NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

A FORMAL MODEL FOR RISK ASSESSMENT IN SOFTWARE PROJECTS

by

Juan Carlos Nogueira de León

September 2000

Dissertation Advisor: Luqi

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE September 2000	3. REPORT TYPE AND DATES COVERED Ph.D. Dissertation

4. TITLE AND SUBTITLE : A Formal Model for Risk Assessment in Software Projects

5. FUNDING NUMBERS
ARO-38690-MA
ARO-40473-MA
DARPA-99-F759

6. AUTHOR(S) Nogueira, Juan C.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

The current state of the art techniques of risk assessment rely on checklists and human expertise. This constitutes a weak approach because different people could arrive at different conclusions from the same scenario. The difficulty of estimating the duration of projects applying evolutionary software processes adds intricacy to the risk assessment problem. This dissertation introduces a formal method to assess the risk and the duration of software projects automatically, based on measurements that can be obtained early in the development process. The method has been designed according to the characteristics of evolutionary software processes, such as efficiency, requirement volatility and complexity. The formal model based on these three indicators estimates the duration and risk of evolutionary software processes. The approach introduces benefits in two fields: a) automation of risk assessment and, b) early estimation methods for evolutionary software processes.

14. SUBJECT TERMS
Risk Assessment, Formal Models, Software Estimation Models, Software Metrics, Project Management.

15. NUMBER OF PAGES
270

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited

A FORMAL MODEL FOR RISK ASSESSMENT IN SOFTWARE PROJECTS

Juan Carlos Nogueira de León
Captain, Navy of Uruguay
B.S., Universidad de la República, 1985
M.S., Universidad O.R.T., 1993

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR IN PHILOSOPHY IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2000

ABSTRACT

The current state of the art techniques of risk assessment rely on checklists and human expertise. This constitutes a weak approach because different people could arrive at different conclusions from the same scenario. The difficulty of estimating the duration of projects applying evolutionary software processes adds intricacy to the risk assessment problem. This dissertation introduces a formal method to assess the risk and the duration of software projects automatically, based on measurements that can be obtained early in the development process. The method has been designed according to the characteristics of evolutionary software processes, such as efficiency, requirement volatility and complexity. The formal model based on these three indicators estimates the duration and risk of evolutionary software processes. The approach introduces benefits in two fields:

a) Automation of risk assessment.

b) Early estimation methods for evolutionary software processes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DEDICATION

This dissertation is dedicated to my family for their love, patience, and support, especially to my daughter, María Victoria, my most fervent supporter.

ACKNOWLEDGMENT

First and foremost, I wish to express my sincere gratitude to the Navy of Uruguay for sending me to this exceptional school, and for all the support provided to my family and myself.

Second, I wish to express my gratitude to Dr. Luqi, my advisor, for her constant guidance and encouragement, and for sharing her knowledge, wisdom, and expertise. I also want to thank the members of my committee for their time, advice, and feedback. To Dr. Valdis Berzins for teaching me to think abstractly. To Dr. Carl R. Jones, my thesis advisor during my M.S., for the confidence he showed by opening the doors for this Ph.D. To Dr. Man Tak Shing for his patience teaching, and for his thoughtful feedback. To Dr. Swapan Bhattacharya for his confidence, support and constant encouragement. To Dr. Mark Nissen for his rich feedback.

I would like to express my gratitude to Dr. Raymond Levitt and his team at Stanford, and Carlos Rivero at Vité for their support and assistance in the study of VitéProject. I want also to thank Dr. Richard Burton and Dr. Borge Obel for their support during the study of OrgCon. I also extend my thanks to Roxanne Zolin for sharing her views about VitéProject and OrgCon. Without the contributions of all these people, this dissertation could not have been completed

Finally, I wish to express my gratitude to Dr. Lawrence Putnam. Dr. Putnam, a former NPS alumni and author of the famous estimation model, gave me all his support when I was in the middle of the intricacies of this research. I corresponded to his company asking for clarification of concepts, without much hope of receiving a response, I must confess. Surprisingly, I did receive a response, and it was from Dr. Putnam himself! The clear insights and experience from someone who is truly knowledgeable about models were very helpful. And again to my surprise, a week later I received a box with his books and papers. Beau geste, Dr. Putnam. I will always be thankful.

I. INTRODUCTION

A. THE IMMATURITY OF SOFTWARE ENGINEERING

"Despite 50 years of progress, the software industry remains years–perhaps decades–short of the mature engineering discipline needed to meet the demands of an information-age society" (Gibbs, 1994). Much research has analyzed this problem using various approaches: formal methods, prototyping, software processes, etc. However, Gibb's assertion still remains true today.

Since the creation of the first computers, tremendous progress has been made in terms of hardware. The introduction of the general-purpose computer has been especially important because of its versatility. The stored program allowed specialized applications created by software. These applications have grown in size and complexity covering numerous human activities. Unfortunately, the ability to build software has not followed the same rate of progress (Hall, 1997. pp xv). Gerald Weinberg said, "to call software development an infant discipline is not a moral judgment, but merely a colorful way to summarize its short history and present existence." (Gilb, 1977. Foreword). Software engineering focuses on planning, developing and maintaining software products. Clearly, the creation of software imposes different challenges from the creation of hardware.

Experience suggests that building and integrating software by mechanically processable formal models leads to cheaper, faster and more reliable products (Luqi, 1997). Software development processes, such as the Hypergraph model for software evolution (Luqi, 1997) and the Spiral model (Boehm, 1988) have improved the state of the art. However, they share a common weakness: risk assessment. This dissertation addresses this issue.

I. INTRODUCTION

A. THE IMMATURITY OF SOFTWARE ENGINEERING

"Despite 50 years of progress, the software industry remains years–perhaps decades–short of the mature engineering discipline needed to meet the demands of an information-age society" (Gibbs, 1994). Much research has analyzed this problem using various approaches: formal methods, prototyping, software processes, etc. However, Gibb's assertion still remains true today.

Since the creation of the first computers, tremendous progress has been made in terms of hardware. The introduction of the general-purpose computer has been especially important because of its versatility. The stored program allowed specialized applications created by software. These applications have grown in size and complexity covering numerous human activities. Unfortunately, the ability to build software has not followed the same rate of progress (Hall, 1997. pp xv). Gerald Weinberg said, "to call software development an infant discipline is not a moral judgment, but merely a colorful way to summarize its short history and present existence." (Gilb, 1977. Foreword). Software engineering focuses on planning, developing and maintaining software products. Clearly, the creation of software imposes different challenges from the creation of hardware.

Experience suggests that building and integrating software by mechanically processable formal models leads to cheaper, faster and more reliable products (Luqi, 1997). Software development processes, such as the Hypergraph model for software evolution (Luqi, 1997) and the Spiral model (Boehm, 1988) have improved the state of the art. However, they share a common weakness: risk assessment. This dissertation addresses this issue.

In the software evolution domain, risk assessment has not been addressed as part of the previous models. In the various enhancements and extensions, the graph model does not include risk assessment steps, hence risk management remains a human-dependent activity that requires expertise. This dissertation provides a formal model to assess the project risk independently of the project manager's experience.

On the evaluation of the spiral model, one of the difficulties mentioned by Boehm was: "Relying on risk-assessment expertise, the spiral model places a great deal of reliance on the ability of software developers to identify and manage sources of project risk. . . Another concern is that a risk-driven specification will also be people-dependent." (Boehm, 1988).

Why has software engineering not reached the maturity level of other forms of engineering? Perhaps the answer lies in the differences between software engineering and other disciplines. One difference is that software engineering is highly dependent on people. A second difference is that software engineering is younger (only forty years versus centuries for civil engineering). A third difference is that the product, software, is intangible. Estimating software's real value at the beginning of the software process is difficult. All these differences create a great deal of uncertainty and equivocality[1] in software development projects.

Many investigations (Boehm, 1989), (Charette, 1997), (Gilb, 1988), (Hall, 1997), (Jones, 1994), (Karolak, 1996), (SEI, 1996) have addressed the problem of risk assessment following the perspective of the traditional disciplines. The tools for risk assessment are guidelines for practices, checklists, taxonomies of risk factors and few metrics. All these methods work fine *if* there is a project manager trained in risk assessment with *enough* experience. Such personnel are very scarce. That is one of the reasons why software engineering is still immature. Software costs are often misestimated

[1] The term equivocality introduced by (Burton & Obel, 1998) means the amount of ignorance about the variables in a system.

and many projects exceed their schedules and budgets. This dissertation provides a formal way to assess the risk of a software project independently of the experience of the decision-maker.

B. RISK AND THE ESTIMATION PROBLEM

As the range and complexity of computer applications have grown, the cost of software development has become the major expense of computer-based systems (Boehm, 1981), (Karolak, 1996). Research shows that in private industry as well as in government environments, schedule and cost overruns are tragically common (Luqi, 1989), (Jones, 1994), (Boehm, 1981). Developing software is still a high-risk activity. Research shows that 45 percent of all the causes for delayed software deliveries are related to organizational issues (vanGenuchten, 1991). Despite the advances in technology and CASE tools, little progress has been done to improve the management of software development projects (Hall, 1997). The acquisition and development communities, both governmental and industrial, lack systematic ways of identifying, communicating and resolving technical uncertainty (SEI, 1996).

This research focuses on software project risk assessment, namely predicting the success of the project. The only ways to evaluate the degree of success of a project are a) to compare planned and actual schedules; b) to compare planned and actual costs; and c) to compare planned and actual product characteristics. Software reliability, an emergent branch of software engineering, has addressed this last part. However, the first two issues have not yet been emphatically addressed.

For many years research has greatly increased our knowledge of software projects. Among such *software laws,* it is known that:

- Manpower and time are not interchangeable (Brooks, 1974).
- Human productivity rates are highly variable, and function and size are highly correlated with errors and duration of the project (Putnam, 1980).

- The majority and most costly errors are introduced during the requirements phase (Boehm 1981).
- Life-cycle manpower patterns follow tailed probability curves (Norden, 1963), (Putnam, 1980, 1992, 1996, 1997), (Boehm, 1981).
- Standards, good practices, guidelines, and heuristics improve the development process (Humphrey, 1989).

Now CASE tools that improve the productivity exist. Macro models also can estimate with different degrees of success the effort and duration of software projects (Albrecht, 1979), (Boehm, 1981, 2000), (Putnam, 1997). What is not available is a model of the internal phenomenology of the software life cycle. Without the knowledge of such a model, scientific risk assessment is almost impossible. This dissertation provides a model to explain the risk of the projects.

In this dissertation, risk is defined as the product of a future outcome times the probability of an occurrence of such an outcome. The outcome could be negative, a loss (this is the general approach that all previous research has applied), but also positive leading to an opportunity.

C. RESEARCH QUESTIONS

The software process is a set of activities with dependency relationships that occur over a certain period of time. From this point of view software projects do not differ from any other type of project. The difference is that for software, the number of activities is uncertain until late in the development process. At the beginning of such a process, a great deal of uncertainty exists. This uncertainty can be reduced through effort, which can be expressed in terms of time and cost. As time goes by, the level of uncertainty usually decreases because more information becomes available. Unfortunately, the main resources (time and budget) also exhibit the same behavior. So project managers, as decision-makers, must choose between making early decisions with a great deal of

uncertainty, or postponing decisions by trading time for information. This leads to the basic research question addressed in this dissertation:

What are the automatically collectable early measures of the software process that indicate project risk?

The concept of early measure is emphasized because recognizing the risks in the early phases increases the probability of success, improving, consequently, the competitive advantage. This research focuses on automatically collectable measures because risk identification should not impose a significant extra workload and must be as objective as possible. This leads to the second question:

How can these measures be used to assess project risk?

D. GENERAL APPROACH

Despite the recent improvements in software processes and automated tools, risk assessment for software projects remains an unstructured problem dependent on human expertise (Boehm, 1988), (Hall, 1997). This dissertation explores ways to transform risk assessment into a structured problem with systematic solutions. Solving the risk assessment problem with indicators measured in the early phases would constitute a great benefit to software engineering. In the requirements phase, changes can be made with the least impact on the budget and schedule. The requirements phase is the crucial stage to assess risk because: a) it has a huge amount of human intervention and communication that can be misunderstood and can be a source of errors; b) errors introduced at this phase are very expensive to correct if they are discovered late; c) the existence of generation tools diminishes the errors in the development process if the requirements are correct; and d) requirements evolve introducing changes and maintenance along the whole life cycle.

Constructing a model to assess risk based on objective measurable parameters that can be automatically collected and analyzed is necessary. One of the goals of this research

is to integrate a risk assessment model with the previous research (Luqi, 1988) on CAPS[2] at NPS. This integration is required in order to capture metrics automatically and to provide project managers with a more complete tool.

Software risk management includes identifying, assessing and mitigating risks. It requires dealing with complexity and assigning scarce resources in the most efficient way. The scope of this dissertation is limited to risk identification and risk assessment. Automated methods can provide major impact in these two areas.

This dissertation studies project risk assessment by dividing it into three classes: *resource risk assessment*, *process risk assessment*, and *product risk assessment*. A dependency between these classes of risk exists. The success of the project depends on the matching between the characteristics of the process, the resources, and the product.

A measure of project success is the probability (p) of developing the required product according to the planned schedule and within the budget applying a certain software process. Consequently, the project risk is the cost associated with its failure times the probability of failure (q = 1 - p). The hypothesis of this dissertation is that the associated probability distribution is skewed to the right.

Creating a set of metrics customized to the characteristics of software evolution including complexity, requirements volatility and efficiency is necessary to support the estimation of risk. The details of such a framework are described in Chapter III. The approach has a fundamental implication: in order to assess risk, one must assess duration of the project and consequently the effort (see Chapter II Section E).

---

[2] CAPS stands for Computer Aided Prototyping System

E. SOFTWARE EVOLUTION FOCUS

Boehm has shown that early parts of the system development cycle, such as requirements and design specifications, are especially prone to errors (Boehm, 1981). As a result, problems originating in the early stages often have a lasting influence on the reliability, safety and cost of the system. Evolutionary prototyping offers an iterative approach to requirements engineering to alleviate the problems of uncertainty, ambiguity and inconsistency inherent in the process. Moreover, prototyping can improve the capture of change in requirements and assumptions during the development process. This effect is particularly observed in projects involving multiple stakeholders with different points of view (Ramesh, 1995), (Conklin, 1988). Software prototyping is characterized by changes in the requirements. Such changes are consequence of the evolution and refinements. For that reason the requirements volatility is one of the metrics used in our model.

Evolutionary driven computer aided software engineering (CASE) tools for computer-aided prototyping provide a logical assessment of the consistency and clarity of requirements and specifications. The use of prototypes facilitates the requirement phase in any type of software projects. Particularly, in real-time applications where severe time constraints impose more challenges, the use of prototypes helps to describe the requirements in a clear, precise, consistent and executable format. Prototypes can be applied to demonstrate system scenarios to the affected parties as a way: a) to collect criticisms and feedback that are sources for new requirements; b) to detect deviations from users' expectations early; c) to trace the evolution of the requirements; d) to improve the communication and integration of the users and the development personnel; and e) to provide early warning of mismatches between proposed software architectures and the conceptual structure of requirements.

The benefits of prototyping are unquestionable. All modern life-cycle models, such as Bohem's spiral, Luqi's graph model, rapid application development (RAD), etc., are based on prototyping. Experience suggests that building and integrating software by mechanically processable formal models lead to cheaper, earlier and more reliable products (Luqi, 1997).

Despite their benefits, evolutionary software processes have two issues that must be solved. The first concern is that they rely on human expertise to identify and assess risk. This dissertation addresses this issue in Chapter VII.

A second concern in the use of prototypes is that they introduce a problem in project planning because of the uncertain number of prototyping cycles required before constructing the product and the amount of complexity that should be covered at each cycle. For the most part, existing project management and estimation techniques are based on linear layouts of activities. CPM and PERT techniques are not well-suited to deal with cycles because they are based on acyclic digraphs. This issue is discussed in Chapter II Section G.

F. CONTRIBUTIONS

The first contribution of this dissertation is the transformation of the unstructured problem of software risk assessment into a structured one (Chapter III). This contribution impacts the software engineering state of the art, but also risk management in general. The use of formal models based on a set of metrics solves the human-dependency issue characteristic the present state of the art in that area. The set of metrics chosen includes the principal characteristics of any evolutionary software process, although it could be expanded to other indicators in future research, includes the principal characteristics of any evolutionary software process.

A second contribution, and perhaps the most important, is the creation of estimation models that can be used from the beginning of the project (Chapter V, Section C; Chapter VI). These models address the requirement issue of the present state of the art estimation models, which rely on an unambiguous and frozen definition of requirements. With the proposed models, project managers will have a decision support tool much earlier in the life cycle. The dissertation shows that commonly used planning techniques, such as Pert, Gantt, and CPM, could result in overly optimistic results when they are applied to communication-intensive projects like software development.

The third contribution is related to software metrics. The risk assessment framework and the estimation models rely on measures that are innovations in the field of software metrics:

- Two complexity metrics specially suited for formal specifications (Chapter V, Section A.3).
- An organizational productivity metric (Chapter V, Section A.2).
- A requirement volatility metric that constitutes by itself a decision support tool (Chapter V, Section A.1).

A fourth contribution addresses the lack of risk assessment in the evolutionary software processes. This dissertation improves the Relational Hypergraph Model by the introduction of a new step addressing risk (Chapter VII, Section A).

G. ORGANIZATION OF DISSERTATION

This dissertation is organized in nine chapters. The introduction is in the present chapter. Chapter II presents relevant theoretical foundations and background on software engineering, software evolution, organizational theory, chaos theory, estimation models, software reliability, and risk management. The conceptual framework of the model is developed in Chapter III. Chapter IV presents the detailed research design. The development of the model, the statistical analysis of the observed results, and the algorithms for estimation are presented in Chapters V and VI. Chapter VII discusses integration with CAPS, introduces an improvement to the evolutionary software process, and discusses the method for risk assessment. Finally, Chapter VIII presents the conclusions and identifies opportunities for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. THEORETICAL FOUNDATION

A. THEORETICAL FOUNDATION FOR SOFTWARE EVOLUTION

1. The Graph Model

The graph model is a data-graph model for evolution that records dependencies and supports automatic project planning, scheduling, and configuration management. The evolution process is represented by a graph that at any given moment models the current, past, and planned future states of the software system.

The graph model has experienced its own evolution process. Luqi introduced the original simple version of the model (Luqi, 1989). Mostov and Luqi refined and elaborated the model (Mostov, 1989, 1990), (Luqi, 1990). Luqi introduced the notion of hypergraph to realize automated software evolution in multidimensional phases (Luqi, 1990). Further refinements, including scheduling and team coordination, were introduced by (Badr, 1993). Conflict resolution of requirements and criticisms were introduced by (Ramesh, 1992) and (Ibrahim, 1996). Luqi extended the graph model to a hypergraph that improved the traceability of dependencies and introduced the concept of hyper-requirements (Luqi, 1997). Finally, Harn extended the model to a relational hypergraph model (Harn, 1998a, Harn, 1998b, Harn, 1998c).

2. Conflict Resolution Model

Evolutionary software development requires a way to resolve the conflicts that could occur between various users' points of view. System design must follow a deliberation process that involves resolving issues that should be addressed to satisfy user requirements. Conklin (Conklin, 1988) applied the IBIS model to resolve conflicts in

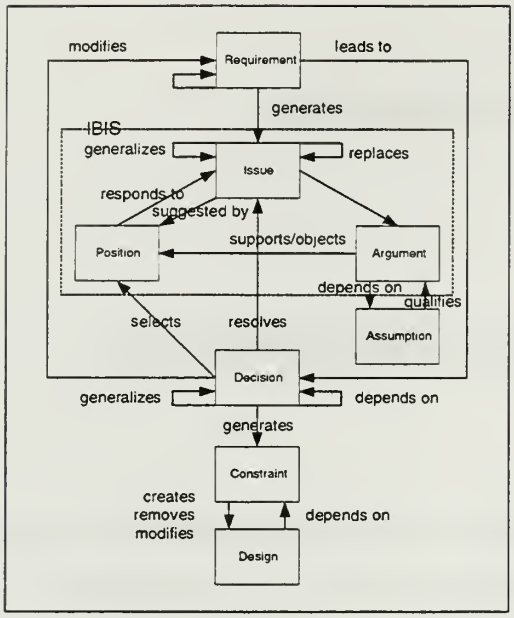


Figure 2.1: REMAP model

requirements. Later Ramesh (Ramesh, 1992) introduced the REMAP model that relates the following concepts:

(1) Requirements represent the goals to be satisfied by the design process.

(2) Issues are questions or concerns that different stakeholders introduce.

(3) Positions are alternatives that address an issue.

(4) Arguments either support or object a position.

(5) Decisions represent the resolution of issues and lead to constraints.

This approach seems to have inspired the Win Win at USC. Win Win is a methodology that aids in the capture, negotiation, and coordination of requirements for large systems. It assumes that a group of people, called stakeholders, has different, probably contradictory, views and positions that require being harmonized in order to elucidate requirements (USC, 2000). Win Win has been implemented for Solaris, SunOS, Linux, and Java.

3. Relational Hypergraph Model

The relational hypergraph model, introduced in (Harn99e), is a formal model for software evolution that incorporates the features of the previous graph models. The hypergraph model (Luqi, 1997) represents the evolution history, as well as the plan for the future, a hypergraph. A hypergraph is a directed graph with hyperedges, which may have multiple input and output nodes. The formal definition of the relational hypergraph model is based on the following definitions:

Definition 1: Directed hypergraph (Harn, 1999f). A directed hypergraph is a tuple H = (N, E, I, O) where N is a set of nodes, E is a set of hyperedges, I is a function giving the set of input nodes of each hyperedge, and O is a function giving the output nodes of each hyperedge.

Definition 2: Path (Harn, 1999f). A path p from node n_1 to node n_k is a sequence of hyperedges $e_1, \ldots, e_{k-1}$ ($k>0$), and a sequence of nodes $n_1, \ldots, n_k$, such that

$n_i \in I(e_i)$ and $n_{i+1} \in O(e_i)$ for $1 \leq i < k$.

Definition 3: Acyclic hypergraph (Harn, 1999f). A hypergraph H = (N, E, I, O) is acyclic if and only if there is no path from any node in H to itself.

Definition 4: Reachable (Harn, 1999f). A set N of nodes is *reachable* from a set R of nodes if and only if there is a path to each node $n \in N$ from some node $r \in R$. A hypergraph H, is reachable from a set R of its nodes, if and only if all its nodes are reachable from R. The *root* of the hypergraph H is a node from which H is reachable. A *leaf* of H is a node from which no other node is reachable.

Definition 5: Composite node and composite edge (Harn, 1999f). A composite node is a set of nodes, and a composite edge is a set of edges.

Definition 6: Hypergraph set (Harn, 1999f). A hypergraph set is the union of nodes and edges of a set of hypergraphs.

Definition 7: Refinement of a composite node (Harn, 1999f). Let H = (N, E, I, O). The refinement of a composite node $n \in N$ is a directed minimal hypergraph $H_m = (N_{in} \cup N_{out}, \{e\}, I, O)$, where the input node set $N_{in} = \{n_1, \ldots, n_n\}$, the output node set $N_{out} = \{n\}$, and the edge set is $\{e\}$. The edge e is called a decomposition edge and relates the node to the nodes in its decomposition.

Definition 8: Opposite hypergraph (Harn, 1999f). Let $H = (N, E, I, O)$ then its opposite hypergraph is $H_{op} = (N, E, O, I)$.

Definition 9: Hyperpath (Harn, 1999f). A hyperpath from N_1 to N_2 in the hypergraph $H = (N, E, I, O)$, is a minimal hypergraph from a set of nodes N_1 to another set of nodes N_2 where $N_1 \subseteq N$ and $N_2 \subseteq N$.

Definition 10: Refinement of a composite edge (Harn, 1999f). Let $H = (N, E, I, O)$. The refinement of a composite edge $e = \{e_1, \ldots, e_n, n_1, \ldots, n_n\}$, where is a hypergraph set of minimal hypergraphs $R = (N_{in} \cup N_{out}, e, I, O)$. $N_{in} = I(e)$, $N_{out} = O(e)$, and $e_1, \ldots, e_n$ are called subedges.

Definition 11: Refinement of a minimal hypergraph (Harn, 1999f). Let $H_m = (N_{in} \cup N_{out}, \{e\}, I, O)$ be a minimal hypergraph. The refinement of a minimal hypergraph is a hypergraph set $R = H_{in} \cup H_{out} \cup H_e$, where H_{in} is a refinement of N_{in}, H_{out} is a refinement of N_{out}, and H_e is a refinement of e. H_m can be viewed as a graph composed of two nodes (N_{in}, N_{out}) and one edge (e) where N_{in} and N_{out} are hypergraphs and e is hyperedge.

Definition 12: Evolutionary hypergraph (Harn, 1999f). An evolutionary hypergraph is a labeled, directed, and acyclic hypergraph $H = (N, E, I, O)$ together with label functions that give component attributes to the nodes and step attributes to the edges.

Definition 13: Top-level evolution step (Harn, 1999f). A hyperedge is called a top-level evolution step if there are no parent evolution steps.

Definition 14: Atomic evolution step (Harn, 1999f). An atomic evolution step is an atomic edge without any refinements.

Definition 15: Top-level evolutionary hypergraph (Harn, 1999f). A top-level evolutionary hypergraph is an evolutionary hypergraph whose edges are top-level evolution steps.

Definition 16: Atomic evolutionary hypergraph (Harn, 1999f). An atomic evolutionary hypergraph is an evolutionary hypergraph with atomic evolution steps as its hyperedge.

Definition 17: Primary input (Harn, 1999f). Primary inputs are previous versions of the output component of an evolutionary step.

Definition 18: Secondary inputs (Harn, 1999f). Secondary inputs are all input components required in an evolutionary step that are not primary inputs.

Definition 19: Primary-input-driven hypergraph (Harn, 1999f). An evolutionary hypergraph is called primary-input-driven if and only if its input nodes are primary inputs.

Definition 20: Secondary-input-driven hypergraph (Harn, 1999f). An evolutionary hypergraph is called secondary-input-driven if and only if its input nodes are secondary inputs.

Definition 21: Relational hypergraph (Harn, 1999f). A relational hypergraph is an evolutionary hypergraph in which the dependency relationships between components and steps can have a hierarchy of specialized interpretations.

Definition 22: Software prototyping demo step (Harn, 1999f). A software prototyping demo step is a step in which the input components are a set of criticisms (C1), a set of programs (P), a set test scenarios (T), and a set of stakeholders (U), producing an output component set of criticisms (C2).

Definition 23: Issue analysis step (Harn, 1999f). A issue analysis step is a step in which the input components are a set of previous issues (J1), a set of stakeholders (U), a set of criticisms (C), producing an output component set of new issues (J2).

Definition 24: Requirement analysis step (Harn, 1999f). A requirement analysis step is a step in which the input components are a set of previous requirements (R1), a set of issues (J), a set of stakeholders (U), producing an output component set of new requirements (R2).

Definition 25: Specification design step (Harn, 1999f). A specification design step is a step in which the input components are a set of previous specifications (S1), a set of stakeholders (U), a set of requirements (R), producing an output component set of new specifications (S2).

Definition 26: Module implementation step (Harn, 1999f). A module implementation step is a step in which the input components are a set of previous modules (M1), a set of stakeholders (U), a set of specifications (S), producing an output component set of new modules (M2).

Definition 27: Program integration step (Harn, 1999f). A program integration step is a step in which the input components are a set of previous programs (P1), a set of stakeholders (U), a set of modules (M), producing an output component set of new programs (P2).

Definition 28: Software product demo step (Harn, 1999f). A software product demo step is a step in which the input components are a set of previous optimizations (K1), a set of stakeholders (U), a set of programs (P), a set of test scenarios (T), producing an output component set of new optimizations (K2).

Definition 29: Software product implementation step (Harn, 1999f). A software product implementation step is a step in which the input components are a set of previous versions of programs (P1), a set of stakeholders (U), a set of optimizations (K), producing an output component set of new programs (P2).

Definition 30: Software prototyping evolution process (Harn, 1999f). A software prototyping evolution step is a hypergraph with a path with the following properties:

- Steps are software prototype or product demo, issue analysis, requirement analysis, specification design, module implementation and program integration.
- Nodes are old version programs, criticisms, issues, requirements, specifications, modules, and new version programs.

Definition 31: Software product generation process (Harn, 1999f). A software product process is a relational hypergraph with a path with the following properties:

- Steps are software prototype or product demo, and program integration.
- Nodes are new version prototypes or old version programs, optimizations, and new version programs.

Definition 32: Software evolution process (Harn, 1999f). A software evolution process is a relational hypergraph with a combined structure of software prototyping evolution processes and software product generation processes.

Definition 33[3]: Top-level relational hypergraph net (Harn, 1999f). A top-level relational hypergraph is a set composed from a set of primary inputs, one or more sets of secondary inputs, and a set of output nodes of a top-level evolution step. (Harn, 1999f) called this is concept SPIDER (Step Processed in Different Entrance Relationships).

[3] This definition is presented for illustration purposes and completeness, but will not be addressed in this dissertation.

Definition 34[4]: Atomic relational hypergraph net (Harn, 1999f). An atomic relational hypergraph is a set composed by a set of primary inputs, one or more sets of secondary inputs, and a set of output nodes to an atomic evolution step. (Harn, 1999f) called this concept an atomic SPIDER.

4. Computer Aided Prototyping System (CAPS)

Computer Aided Prototyping System (CAPS) is a CASE tool that provides a collection of techniques and languages for computer-aided prototyping, including logical assessment of the consistency and clarity of requirements and specifications. CAPS methods involve the use of real-time constraints and abstract modeling to describe the requirements in a clear, precise, consistent and executable format. Prototypes can be applied to demonstrate system scenarios to the affected parties as a way to elucidate requirements.

Real-time systems present special difficulties in terms of requirement engineering. Some requirements are difficult to provide for the user, and difficult to determine for the analysts. The best way to discover these hidden requirements is via prototyping. CAPS is a tool specially suited for this task. It has a graphical, easy to understand, interface that maps to a specification language, which in turns generates Ada code. The main components of CAPS are:

(a) The prototype system description language (PSDL).

(b) User interface based on a graphic editor with a palette of objects that include operators, inputs, outputs, data flows and operator loops.

(c) The software database system provides a repository for reusable PSDL components. A search engine helps the designer to find reusable components.

(d) The execution support system consists of a translator, scheduling mechanisms, execution monitors, and a debugger.

---

[4] This definition is presented for illustration purposes and completeness, but will not be addressed in this dissertation.

The prototyping process consists of prototype construction and modification (evolution) based on evolving requirements and code generation. Both construction and modification are exploratory activities with a common target: to satisfy multiple users with different and often conflicting points of view. Requirement engineering is a consensus-driven activity in which mechanisms for conflict resolution and traceability of requirement evolution represent critical success factors.

PSDL is based on data flow under real-time constraints and uses an enhanced data flow diagram that includes non-procedural control and timing constraints. PSDL serves as an executable prototyping language at a specification or design level. The user interface contains a graphic editor, a browser to view reusable components, and an expert system that provides the capability to generate English text descriptions of PSDL specifications.

The software database system provides the repository facilities for reusable components, and for control of versions. The execution support system consists of a translator that generates code that binds the reusable components, scheduling mechanisms, and a debugger.

The model views a software evolution process as a partially ordered set of steps. Steps represent activities required to produce the system. A step has states that reflect the dynamic progression of the activity from the moment the step is proposed to the moment it is completed or abandoned.

5. Conclusions about the Relational Hypergraph Model and CAPS

The precedent definitions constitute the formal specification of the relational hypergraph model. They constitute a framework to support the software evolution processes. CAPS is based in part on these definitions. The relational hypergraph model is supposed to support any software development process, including processes with risk assessment steps. However, the model has not been previously applied to such a process,

and has not been specialized to include features needed just for risk assessment. This issue creates a human dependency in risk assessment. Despite this limitation, the model can be extended to support automated risk assessment. Solving this issue is one of the goals of this dissertation (see Chapter VII).

B. THEORETICAL FOUNDATION FOR RISK MANAGEMENT

The research on risk and risk management is very extensive. It comprises operational research, project management, software engineering, and software reliability. Operational research provides the theoretical foundation to describe and analyze risk. Project management, software engineering, and software reliability apply the theory. This research narrows the problem to software, specifically to the software engineering domain.

1. Risk and Uncertainty

Developing software is still a high-risk activity. Despite the advances in technology and CASE tools, little has improved the management of software development projects. The acquisition and development communities, both governmental and industrial, lack a systematic way to identify, communicate and resolve technical uncertainties (SEI, 1996). Research shows that 45 percent of all delayed software deliveries are related to organization issues (vanGenuchten, 1991). Software is the main expense in computer systems (Boehm, 1981), (Karolak, 1996). Besides the improvements in tools and methodologies, there is little evidence of success in improving the process of moving from the concept to the product. A study published by the Stadish Group reveals that the number of software projects that fail has dropped from 40% in 1997 to 26% in 1999. However, the percentage of projects with cost and schedule overruns rose from 33% in 1997 to 46% in 1999 (Reel, 1999).

Part of the problem is misinterpreting the importance of risk management. It is usually and incorrectly viewed as an additional activity layered on the assigned work, or worse, as an outside activity that is not part of the software process (Hall, 1997), (Karolak, 1996).

A second source of problems in risk management is the lack of tools (Karolak, 1996). The main reason for this lack of tools is that risk assessment is apparently an unstructured problem. Structured problems involve routine and repetitive problems for which a standard solution exists. Unstructured problems require decision-making based on a three-phase method (intelligence, design, choice) (Turban & Aronson, 1998). An unstructured problem is one in which none of the three phases is structured.

Risk management is highly biased by a manager's perceptions and characteristics, which are difficult to represent by an algorithm. Depending on the decision-maker's risk behavior, he or she can decide early with little information, or can postpone the decision, gaining time to obtain more information, but losing some control.

A third source of the risk management problem is the confusion created by the informal use of terms. Often, the software engineering community (and most parts of the project management community (Wideman, 1992)) use the term "risk" casually. This term is often used to describe different concepts. It is erroneously used as a synonym of "uncertainty" and "threat" (SEI, 1996), (Hall, 1997), (Karolak, 1996).

In this research the term "risk" is reserved to indicate the probabilistic outcome of a succession of states of nature, and the term "threat" is used to identify the dangers that can occur. Generally, software risk is viewed as a measure of the likelihood of an unsatisfactory outcome and an expected loss affecting the software from different points of view: project, process, and product (Hall, 1997), (SEI, 1996). However, this definition of risk is misleading because it confounds the concepts of risk and uncertainty. In general, most parts of the decision-making in software processes is under uncertainty rather than

under risk. The definition of risk presented in Chapter I stated that risk is the product of the value of an outcome times its probability of occurrence. This outcome could be positive (gain) or negative (loss). This abstraction permits one to address not only the classical risk management issue, but also to discover opportunities leading to competitive advantage. Now let's discuss briefly the decision-making environments in order to clarify these concepts.

Three possible situations exist in any decision context: certainty, risk and uncertainty. Decisions under "certainty" occur when the decision-maker knows the exact consequence of each alternative or decision. In this case the decision process is very simple: the alternative with the best outcome is chosen. However, this is a rarity.

Usually the decision-maker does not have a complete picture of the future, but knows the probability of occurrences of the various possible states of nature. In this case the decision-making is under risk, and many techniques can be addressed to support the decision: expected monetary value, expected value of perfect information, opportunity loss, and sensitivity analysis, among others (Render, 1997). All these methods rely on the huge hypothesis of knowing the exact probability for each state of nature.

A completely different situation is when the decision-maker does not have precise information about the probability distribution of the different states of nature. In these cases a completely different set of techniques must be applied to support the decision-making process: maximin, minimax, Laplace, Hurwicz, or minimax regret (Render, 1997). These methods are explained in Section B.2.

The distinction between risk and uncertainty is important for decision making because it leads to drastically different approaches to risk assessment:

(a) **Assessing Software Risk by Measuring Reliability.** In this case the decision-making is under risk. However, uncertainty exists even using probabilistic models.

This is created by uncertainties in parameter values, uncertainties in modeling, and ambiguities in the degree of completeness such as: (Baybutt, 1989)

- Ambiguities in parameter values are consequences of the need to estimate parameter values from data. These ambiguities arise because the available data is usually incomplete and because the analyst, depending on incomplete knowledge, makes inferences.
- Deficiencies of the model in representing reality.
- Completeness ambiguities are introduced by the analyst's inability to evaluate exhaustively all contributions to risk.

(b) **Assessing Software Risk Using Practices and Guidelines** (SEI, 1996). In this case there is no probabilistic model to rely on, hence the decision-making is under uncertainty.

It follows, as previously stated, that most software-managers' decisions are made under uncertainty. Three groups of researchers view the issue from different angles. The researchers who follow the probabilistic approach have successfully assessed the reliability of the product (Lyu, 1995), (Schneidewind, 1975), (Musa, 1998). However, this approach assesses software reliability when it is too late to economically correct possible faults because the product is complete or almost complete. This approach is discussed in Section E.

Other researchers assess the risk from the beginning, in parallel with the development process. However, in this case, the approach is less rigorous and unstructured. Basically the proposals are lists of practices and checklists (SEI, 1996), (Hall, 1997) or scoring techniques (Karolak, 1996). Paradoxically, SEI defines software technical risk as a measure of the probability and severity of adverse effects in developing software that does not meet its intended functions and performance requirements (SEI, 1996). However, the term "probability" in this case is misleading because the applicable probability distribution is unknown. This approach is discussed in Section C.

(c) **Assessing Software Risk by Using Estimation Models**. A third group of researchers focus mainly on the estimation of effort and time that has characteristics of both previous groups. This approach is tangentially related to risk and will be discussed in Section D.

2. Decision under Uncertainty

Quite frequently decision-makers must use incomplete information. Particularly, the problem of decision-making under uncertainty involves choosing among a set of alternatives under the following conditions:

- The outcome of each course of action depends on several possible states of nature.
- The outcome for each alternative under each state of nature is known.
- The probability of occurrence of each state of nature is unknown or known only very roughly.

When the probability of occurrence of each state of nature is unknown or cannot be assessed, then the following five techniques can be applied:

(1) *Maximax Criterion*. This criterion implies an optimistic vision of the future. The method consists of choosing the alternative that maximizes the outcome for every state of nature.

(2) *Maximin Criterion*. This method finds the alternative that maximizes the minimum outcome. It represents a pessimistic approach.

(3) *Laplace Criterion*. This method uses equal probabilities for each state of nature and then computes the outcomes for each state of nature, choosing the highest outcome.

(4) *Criterion of Realism*. This method is also known as Hurwicz Criterion (Render, 1997). It is a compromise between an optimistic and a pessimistic decision. The decision-maker must choose a coefficient of realism α between 0 and 1. This

coefficient is applied to the most favorable state of nature outcome, and (1 - α) is applied to the outcome of the most unfavorable state of nature. The alternative with the higher weighted sum is chosen.

(5) *Minimax Criterion*. This method is based on opportunity loss. The opportunity loss refers to the loss that would be suffered by making the wrong decision (Render, 1997). The method finds the alternative that minimizes the maximum opportunity loss within the alternatives.

There is no objective way to decide which of these approaches is the best. The final result comes from the comparison between the decision made and the reality. The choice between them relies on the preferences of the decision-maker and his acceptance or avoidance of risk.

3. Subjective Probabilities and Utility Theory

Another way to deal with uncertainty situations is to subjectively estimate the probabilities of occurrence of the different states of nature. This approach is easy to implement but requires a great deal of experience to judge the success probability of each alternative. Group consensus techniques (like the Delphi Method) are usually quite helpful in such situations (Marshall, 1995), (Putnam, 1992).

When a decision is made under risk, that is when the probability distribution function of the states of nature is known, it is possible to support the decision process using decision trees. In general, decision trees based on the expected monetary value (EMV) could only lead to bad decisions in many cases. There are many situations in which a linear payoff function is unable to represent the behavior of people (Marshall, 1995). These are the two reasons to study *utility theory*. In practice, historical data can be analyzed to obtain an objective estimate of the outcomes. But in situations, especially those that incorporate management decisions, historical data could be irrelevant. The judgments and beliefs of the decision-makers may be more important that estimating

relevant probabilities (Marshall, 1995). Before describing utility theory in detail, two definitions are required:

> The indifference probability for a decision problem between a risky venture and a riskless alternative with given known results is that probability of success in the risky venture for which the decision-maker is indifferent to the two alternatives. (Marshall, 1995).
> The certainty equivalent to a risky venture is the least amount the decision-maker would have to obtain for certain by choosing the riskless alternative. (Marshall, 1995).

In many situations the indifference probability and the certainty equivalent would have different values for different people. The differences reflect various behaviors toward risk. Utility assessment assigns a utility of 0 to the worst outcome and a utility of 1 to the best outcome. All other outcomes have a utility value between 0 and 1. When two or more alternatives are equally attractive (or unattractive), that is the decision-maker is indifferent, then their utility value should be the same. The problem is to find the probability that makes the decision-maker indifferent.

Until now, only one-attribute decision-making problems have been considered. A more general scenario would have many attributes for measuring the decision. Often, these attributes conflict with each other, hence optimizing one attribute results in suboptimizing others. Thus, using trade-offs to resolve such conflicts is necessary. A common approach to solving multiattribute problems is to combine the different measures into a single numeric measure. The problem can then be treated as single attribute problem (Marshall, 1995). In many decision problems, establishing measurement criteria is highly challenging, particularly when the decisions are not at the operational level. At the operational level, decisions can be measured in terms of lines of code or function points. However, at the project management level, the effectiveness of a decision could be measured in terms of quality, stability, marketing impact, etc. In such cases, multiattribute utility theory should be applied (Fig. 2.2).

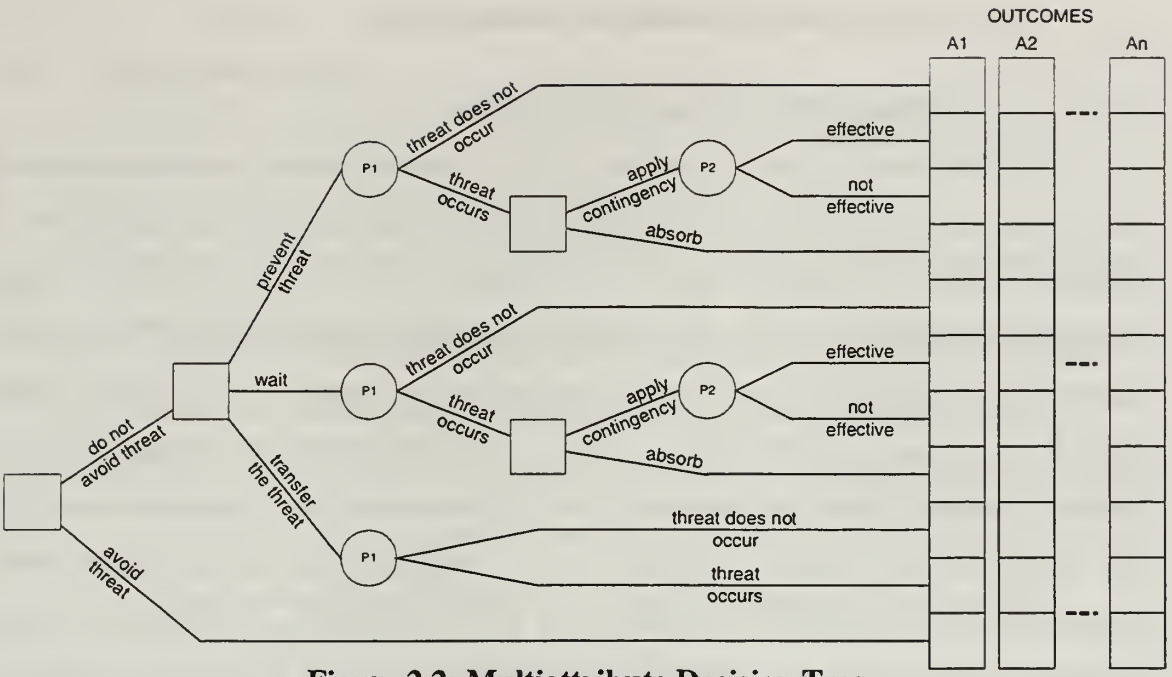


Figure 2.2: Multiattribute Decision Tree

The decision-maker must provide his estimation of return for each attribute related to the decision, as a vector R = (R1, R2, ..., Rn). The decision-maker must also introduce his preferences as a weight vector W = (W1, W2, ..., Wn). The outcomes of each attribute are given by Ai, such:

$$Ai = Wi * Ri$$

$$\text{where } \sum_{i=0}^{n} Wi = 1$$

The outcome for each alternative is then calculated as a function of the sum of the attributes (A1, A2, ..., An) converted to a value between 0 and 1, where 1 is given to the best outcome and 0 to the worst.

C. SOFTWARE ENGINEERING FOUNDATIONS

1. Software Engineering Institute (SEI)

The Software Engineering Institute (SEI), at Carnegie Mellon, relies on improving the process as a way to improve the products and to diminish risks. This philosophy is particularly clear in a guideline created at the request of the USAF by the SEI and Mitre Corporation (Humphrey, 1987). The document describes a method to assess the software engineering capabilities of contractors. The guideline stated that the quality of the product depends on the quality of the process, which in turn depends on the technology used to support it, which depends on the maturity level of the organization. Hence, by transitivity, the quality of the product depends on the maturity level of the organization. Consequently, by assessing the maturity of the organization, one can estimate the attributes of the product. SEI proposes a three-dimensional vision of the risk management process (SEI, 1996). This vision is as follows:

(a) The temporal dimension that includes the micro perspective, that is from the point of view of the project, and the macro perspective, which covers the complete life cycle.

(b) The methodological dimension that includes practices (software risk evaluation (SRE), continuous risk management (CRM) and team-risk management (TRM)), and basic constructs including the SEI's risk taxonomy.

(c) The human dimension that considers the perspectives of the individual, the team, the management and the stakeholder.

The SEI approach to risk assessment uses a risk taxonomy questionnaire to ensure that all risk areas are systematically addressed. The complete taxonomy can be reached in (SEI96). Table 2.1 presents a brief summary to show the characteristics analyzed.

Table 2.1: SEI's Taxonomy of Risks (SEI, 1996)

1.	***Product engineering***
1.1.	*Requirements (stability, completeness, clarity, validity, feasibility, precedent, and scale).*
1.2.	*Design (functionality, interfaces, performance, testability, hardware constraints, and non-developmental software).*
1.3.	*Code and unit test (feasibility, testing, coding/implementation).*
1.4.	*Integration and test (environment, product, system).*
1.5.	*Engineering specialties (maintainability, reliability, safety, security, human factors, and specifications).*
2.	***Development environment***
2.1.	*Development process (formality, suitability, process control, familiarity, and product control).*
2.2.	*Development system (capacity, suitability, usability, familiarity, reliability, system support, and deliverability).*
2.3.	*Management process (planning, project organization, management experience, program interfaces).*
2.4.	*Management methods (monitoring, personnel management, quality assurance, and configuration management).*
2.5.	*Work environment (quality attitude, cooperation, communication, and morale).*
3.	***Program constraints***
3.1.	*Resources (schedule, staff, budget, and facilities).*
3.2.	*Contract (type of contract, restrictions, and dependencies).*
3.3.	*Program interfaces (customer, associate contractors, subcontractors, prime contractor, corporate management, vendors, and politics).*

The SEI approach however presents the following problems:

(a) Many of the items covered by this taxonomy are highly subjective and difficult to express in terms of equations. How to measure politics? How to measure with confidence the morale? The only way is to use qualitative measures that have inherent subjectivity.

(b) Many of the items are covered more than once, for instance, human factors, work environment, and budget seem to be highly related.

(c) The guidelines are sets of heuristics and good practices which impact the success of the project and depend on human experience.

Consequently, this approach relies on the ability of the project manager using the checklist. An expert is required to assess the risk.

2. Hall

Elaine Hall's method for managing risk (Hall, 1997) is derived from the SEI model. In her opinion, four major critical success factors are responsible for risk management: *People, Process, Infrastructure,* and *Implementation* (P^2I^2).

(a) People participate in risk management by implementing the processes according to the plans, by detecting problems, communicating issues and introducing uncertainties in their work. People at all levels need to be trained, involved, and motivated in risk management.

(b) Process must transform uncertainties into risks. The transformation is based on identifying the sources of risk, analyzing the risk based on some established criteria, planning alternative strategies for risk resolution, tracking the risk metrics, and resolving the risk triggering action plans. Unfortunately, how to perform this transformation (that is the key problem), is not addressed in (Hall, 1997) and neither in (SEI, 1996).

(c) Infrastructure establishes the culture that supports risk management.

(d) Implementation is the execution of the plans, assigning responsibilities, authorities, tools and methods.

In Hall's method, checklists based on SEI taxonomy, work breakdown decomposition, meetings, reviews and surveys are the tools for risk identification. All these tools are human dependant and highly unstructured. Hence, the method is very difficult to automate. However, Hall emphasizes the use of metrics to identify occurrence of risks, such as progress in milestones, size (LOC), change (requirements added, changed, deleted), quality (number of defects), staff (turnover) and risk exposure. Risk analysis, risk planning, risk tracking and risk resolution are based on planning, and on a set of resolution techniques and tools inherited from SEI's model. Hall's approach has the same problems as SEI's model.

3. Charette

Charette introduced the concept of risk management in maintenance (Charette, 1997). The author states that during maintenance, risk management is more difficult than during development. First, maintenance projects provide more opportunities for risk and less freedom to mitigate risk as a consequence of the previous version of the system. Second, risk management in maintenance requires more attention to customer-related issues. The approach is based on SEI's taxonomy as the tool to identify treats and SEI's software risk evaluation process to assess the risk. Charette's approach has the same problems that were previously addressed about SEI's model. The method relies on human experience.

4. Jones

During the 60's and the 70's IBM has focused significantly on software processes. Many technologies were invented in IBM's laboratories: HIPO diagrams, joint application design, formal inspections, structured walkthroughs, integrated cost and estimation tools, and formal specifications. Significantly, CMM has characteristics that can be traced back to IBM when Watts Humphrey was at IBM. Neither SEI's CMM or Software Productivity Research (SPR) (Jones, 1994) addresses how to solve the problems of estimation. SPR is a software process introduced by Capers Jones that has some characteristics very similar to CMM. Jones and Humphrey were working at IBM during the seventies, so it is not surprising that both models have common characteristics. As an example the five-level scale of CMM corresponds to the five-scale of SPR. (Jones, 1994) observed those significant risks are not the same across all software domains. He introduced six categories of software projects with different kinds of risks. Table 2.2 shows the percentage of projects at risk for each category. Note that the table is ordered showing on the top the risk factors more common for all the projects categories. Jones stated that the ten most serious risk factors observed in the SPR assessments are:

(1) *Inaccurate metrics.* The generalized use of LOC as a productivity metric introduces errors because the differences in the languages and

programming styles. Counting LOC does not address the complexity involved in recursion nor object-oriented paradigm. LOC is very difficult to estimate during the requirements. Albrecht addressed this problem with the introduction of function points. However, recently Kitchenham, Kemerer and others have criticized this metric. This issue is discussed in Chapter V.

(2) *Inadequate measurement.* Data collection is not always correctly done, even in the case of cost collection. One major leak in terms of cost is the work of end users.

(3) *Time pressure introduced by irrational schedules or by continuously changing requirements.* This second factor is more intense as the complexity of the systems grows. Projects with more than 1000 function points are most likely to experience this problem.

(4) *Management weaknesses due to lack of education in estimation, planning, measuring and assessment.*

(5) *Inaccuracies in cost estimation.* Despite the numerous commercial software tools available, the use of estimation tools is not generalized.

(6) *Naive belief that moving to a new technology will create improvements in productivity or quality.*

(7) *Late requirements.* Even with the availability methodologies like prototyping, JAD or QFD, and metrics like function points or feature points, which permit to understand the impact of changes, late requirements continue to be a major threat.

(8) *Low quality.* The current average of defects per function point in U.S. is 5 defects per function point.

(9) *Low productivity.* The current U.S. average for military projects is about 3 function points per man-month. For MIS the productivity is about 8 function points per man-month.

(10) *Cancellation of projects is directly proportional to their size.* This particularly critical above 10,000 function points or 1 million LOC.

Table 2.2: Jones' Top Risk Factors (Jones, 1994)

Risk factor	MIS	Embedded	COTS	Military	Outsource	End-user
Low quality	60%	50%	55%	45%		65%
Schedule	65%	70%	50%	75%		
Creeping user requirements	80%			70%	45%	
Excessive paperwork		60%		90%		
Cost estimates	55%	65%				
Low productivity				85%		
Non-transferable applications						80%
Inadequate documentation			70%			
High maintenance costs					60%	
Inadequate configuration control	50%					
Friction between personnel					50%	
Acceptance criteria					30%	20%
Maintenance problems						50%
Redundant applications						50%
Competitors			45%			
Cancellation		35%				
Litigation expense			30%			
Legal ownership of deliverables					20%	

Jones reveals some common threats characteristics of different types of software projects. The impact of paperwork and low productivity in DoD projects is particularly significant. The caveat of this work is that it does not provide a method to manage risk because it relies on the experience of the project manager to make the right decisions.

5. Karolak

Karolak introduced a classification scheme that divides the risk in three software-risk elements: Technical, Cost and Schedule (Karolak, 1996). This model uses a subjective Bayesian probability approach to assess software risks. Each of the three software risk elements are influenced by ten risk factors listed in Table 2.3:

Table 2.3: Karolak's Scheme (Karolak, 1996)

	Software Risk Elements		
Software Risk Factor	Technical	Cost	Schedule
Organization (a)	LOW	HIGH	HIGH
Estimation (b)	LOW	HIGH	HIGH
Monitoring (c)	MEDIUM	HIGH	HIGH
Methodology (d)	MEDIUM	HIGH	HIGH
Tools (e)	MEDIUM	MEDIUM	MEDIUM
Risk culture (f)	HIGH	MEDIUM	MEDIUM
Usability (g)	HIGH	LOW	LOW
Correctness (h)	HIGH	LOW	LOW
Reliability (i)	HIGH	LOW	LOW
Personnel (j)	HIGH	HIGH	HIGH

(a) "Organization" addresses risks associated with the maturity of the organization structure, functions, management and communications.
(b) "Estimation" addresses the risks associated with inaccuracies in estimating resources, schedules and costs.
(c) "Monitoring" refers to risks associated with identifying problems.
(d) "Methodology" addresses the risks associated with the lack of formal methodology and standards.
(e) "Tools" refers to the risks associated with the development tools.
(f) "Risk culture" addresses the characteristics of the management decision-making style.
(g) "Usability" refers to risks associated to the software product after it is delivered.
(h) "Correctness" addresses to the risks associated with compliance with requirements after the delivery.
(i) "Reliability" refers to the risks of failures after the delivery.
(j) "Personnel" includes the risks associated with the knowledge and skills of the development team.

The key element to identify and measure risks on Karolak's approach is a questionnaire used to evaluate the risk factors (81 questions: organization 8, estimation 7,

monitoring 7, methodology 7, tools 9, risk culture 11, usability 6, correctness 9, reliability 12, and personnel 12). The answer for each question in a number between 0 and 1, where 0 represents none and 1 represents all. The main contribution of this model is that it can be automated; indeed Karolak developed a tool called SERIM (Software Engineering Risk Model). However, the problem with this approach is that even though the tool provides support, human experience is still required as the key factor to identify risks.

6. Project Management Institute (PMI)

The Project Management Institute (PMI) introduced a methodology for risk management (Wideman, 1992) generalized for any kind of projects. The method is based on four phases: risk identification, risk assessment, risk response, and documentation. Risk identification follows an informal approach based on taxonomies, expert's opinions and workgroup techniques. The assessment phase may range from subjective evaluation to the use of metrics. This phase also includes the analysis of impact. In this model there are two planning activities: response planning, and contingency planning; and three typical risk response strategies: avoidance, deflection, and absorption. PMI uses the term "risk" to denote two different concepts: the probability of occurrence of a threat and the threat itself. Another terminology issue in this approach is the use of the term "risk" in scenarios in which decisions are made under uncertainty rather than risk. The approach is too general to be useful in software engineering.

7. Mitre Corporation

The Mitre Corporation developed a Web application (RAMP) to capture risk management experience and retrieve experiences from other projects and advice. The user introduces the characteristics of his project in a static HTML form. A query is launched over the RAMP databases creating a dynamic HTML form with a set of projects with similar characteristics. The user can select one or more of these projects and a second script retrieves risks from the database. The result of this second query is a report

containing links to the applicable documents (Garvey, 1997). This approach helps the decision-maker by providing him related documents about similar projects, but it does not release the need of human experience to manage risk.

8. Rockwell

At Rockwell, an improvement on communicating risks more effectively provided the following benefits: predictable program performance, better reviews, improved process, and improved management practices. Three key elements are the reason for successful risk management at Rockwell: repeatable process, widespread access to adequate knowledge and functional behavior (defined as human factors).

Functional behavior implies human interactions, motivations and incentives, perceptions and perspectives, communication and consensus, and decision making and risk tolerance. (Gemmer, 1997) identified the following functional behaviors:

- manage risk as an asset
- treat decision making as a skill
- actively seek risk information
- seek diversity in perspectives and information sources
- minimize uncertainty on time, control and information
- recognize and minimize bias in perceiving risk
- plan for multiple futures
- be proactive
- improve the decision-making skills
- reward those who identify and manage risks early

Gemmer identified the following causes for risks: uncertainty in time, uncertainty in control, and uncertainty in information. Risk management is usually an uncertainty scenario characterized by: a) uncertainty in the impact or consequence, b) a time frame to

prevent or mitigate risks exists, c) a coupling or domino effect exists, d) uncertainty about the probability distribution function exists (Gemmer, 1997).

9. Boehm

Boehm has been studying the problem of risk management for more than one decade. His contributions to the area are notable. He introduced the importance of verification and validation of software requirements and design specifications during the early phases of the project as a way to mitigate risk (Boehm, 1984). Such activities include: *completeness*, *consistency*, *feasibility*, and *testability of the specifications*. Completeness implies that all the documents and references exist and that there are no missing items, functions or products. Consistency is both internal and external, and implies traceability. Feasibility requires validation that the project can be achieved with the actual resources, that it will satisfy the users' needs, that it will be maintainable, and that its risk has been estimated. Testability requires unambiguous and quantitative specifications.

Boehm introduced the Spiral Model (Boehm, 1988) as a substitute to the Royce's Waterfall model. The Spiral model was the first software process in which risk assessment was the driven factor. The author recognized however that numerous difficulties in applying his model exist:

- matching the evolving process with contracts
- relying on risk-assessment expertise, the model is people dependent in terms of identification, management and risk-driven specification
- the need of further elaboration in the spiral steps (Boehm88)
- ambiguities about how to initiate, terminate and iterate within the spiral
- complexities in handling incremental development, such as refinements from previous versions
- difficulties in formalize processes
- some steps were more complex than were envisioned (Boehm, 1988a)

Boehm introduced a method for risk management that is summarized on Table 2.4. Risk management is divided in two families of activities: risk assessment and risk control (Boehm, 1989) and (Boehm, 1991).

Table 2.4: Boehm's Classification (Boehm, 1991)

A. Risk Assessment is decomposed into:

(1) Risk identification by use of checklists, decision driver analysis, assumption analysis, and decomposition.

a. Checklist (top 10 risks)

- Personnel shortfalls
- Unrealistic schedules
- Requirement risks
- Developing the wrong functionality
- Developing the wrong user interface
- Developing extra functionality not essential or with marginal usefulness
- Continuous stream of requirement changes
- Problems in external components
- Problems in external tasks
- Performance shortfalls. Straining computer science capabilities (trying to do more than the possibilities of the state of the art technology): distributed processing, AI, human-machine interface, algorithm speed and accuracy, computer security, reliability and fault tolerance.

b. Decision driver analysis:

- Politically driven decisions
- Marketing driven decisions
- Applying the wrong solution to the problem because there exist compromises or preferences
- Short-term versus long-term decisions

c. Assumption analysis

- Comparison with previous experience
- Pessimistic approach (Murphy's Law)

d. Decomposition

- Pareto 80-20 phenomena
- Task dependencies (high fan-in implies risk: if anything slips, the project aborts. High fan-out also implies risk: if the precondition slips, then the effect is in many parts of the project)
- Uncertainty areas in the plan

(2) Risk Analysis:

a. Decision trees
b. Network analysis using PERT and probabilistic network analysis
c. Cost risk analysis using COCOMO, Putnam or other estimation tool for effort and duration
d. Automated analysis tools (PROMAP, PROSIM, RISNET, SLAM, Opera/Open Plan, PRISM, REP)

(3) Risk Prioritization:

a. Assess the risk probabilities from historical data, Delphi or other group technique
b. Deal with compound risks
c. Deal with triggered risks (dominoes effect).

B. Risk control is decomposed into:

(1) Planning
(2) Resolution
(3) Monitoring (milestone tracking and top-10 risk tracking)

Boehm warned that current approaches to the software process may have a tendency to create high-risk commitments. "The waterfall model tempts to over promise software capabilities in contractually binding requirements specifications before analyzing the implications. The evolutionary development makes too easy to introduce new ideas and requirements that can lead to a disaster." (Boehm, 1991). In an article coauthored with De Marco, they showed a pessimistic and pragmatic view stating "doing software risk management makes good sense, but talking about it can expose you to legal liabilities. If a software product fails, the existence of a formal risk plan that acknowledges the possibility of such a failure could complicate and even compromise the producer's legal position." (Boehm, 1997).

Boehm's contributions to risk management are multiple. This research picked the most important ones, such as the Spiral model, his analysis of the activities required for risk management, and his risk management method. Due to its relevance, a separate section includes a discussion about COCOMO. Despite his contributions, Boehm recognizes that the issue of relying on humans to assess risk remains unsolved. The use of

checklists, decision driver analysis, assumption analysis, and decomposition is not enough to automate risk identification and assessment.

10. McFarlan

McFarlan introduced a model to assess risk on information system projects based on a three-dimensional checklist covering the three major dimensions which influence the risk inherent in a project: (McFarlan, 1974)

- project size in terms of budget, staffing levels, elapsed time and number of departments affected
- experience with the technology
- project structure in terms of defining the tasks and deliverables

The importance of his contribution resides in the identification of different facets on software projects. This model relies on checklists and in the experience of the decision-maker to evaluate risk.

11. Gilb

In his classical text on Software Engineering Management Gilb presented a set of principles or rules of engagement with risk (Gilb, 1988). The approach is informal. Gilb's principles are heuristics and were the state of the art at that time. His work was included because he was a pioneer in recognizing the problem and in recognizing the need to be proactive.

12. USAF

(USAF, 1988) defines risk as the probability at a given point in a system's life cycle that the predicted goals could not be achieved with the available resources. Due to the high degree of uncertainty, high precision is not useful during the early phases. As the

system progresses, the uncertainty is transformed into risk; therefore, higher precision is required. The USAF introduced a method to abate risk based on checklists and estimating the probability of occurrence and effects. They decompose the software risk in four dimensions: performance, support or maintainability, cost and schedule. The effects on the project are categorized into catastrophic, critical, marginal and negligible. The four risk dimensions are measured in terms of their probability of occurrence and their effect according to Table 2.5.

Table 2.5: USAF Scheme for Risk

Prob. / Impact	1.0 - 0.7 Frequent	0.7 - 0.4 Probable	0.4 - 0.0 Improbable	0.0 Impossible
Catastrophic	HIGH			
Critical		MODERATE		NONE
Marginal				
Negligible			LOW	

The USAF method is very simple and robust. However, it is informal, relying on checklists and the experience of the evaluator.

D. ESTIMATION MODELS

This section presents three models to estimate effort and duration in software projects: COCOMO, Putnam and function points. These estimation models are important because they constitute a preliminary approach to assess risk.

1. The COCOMO Family

Constructive Cost Model (COCOMO) introduced by (Boehm, 1981) is a family of models constituted by Basic, Intermediate and Detailed COCOMO. Basic COCOMO is

an easy to calculate model applicable to small to medium software projects. Intermediate COCOMO is based on the Basic model and includes effort adjustment factors. The detailed COCOMO explains the influence of these additional factors on individual project phases. These earlier models are known as COCOMO 81.

Projects are classified into three categories:

- *organic* which are characterized by small size, small teams and low environmental noise
- *embedded* characterized by strong complex coupling with hardware or other kind of tight constraints like real time systems
- *semidetached* which are intermediate between the previous two categories.

The details of the model can be found in (Boehm, 1981), but it is important to highlight the following assumptions that show the optimistic bias of the model.

- The model assumes that the requirements are defined and that they will remain unchanged.
- The development period according to COCOMO 81 starts at the beginning of the design phase. The requirements phase is not covered.
- The estimation covers only the direct-charged labor. In other words the effort applied in meetings and communication is not considered.
- The model assumes that a man-month is 152 hours of working time.
- The model assumes that the project will have good management.

The input parameter for COCOMO 81 is the size estimation in KLOC, which constitutes a drawback because of the difficulty of predicting the size during early stages. COCOMO II addresses the problem of size estimation introducing a more abstract indicator of size called object points (a variation of function points also called application points). Object points can be used as input for the model in the case of small projects that can be developed in a few months. For bigger projects the input parameter for the model

is an estimate of the size in lines of code. Object points can be used to derive an estimation of the lines of code. (Boehm et al, 2000). This model was calibrated using 83 projects (Chulani, et al., 1999), (Boehm, 2000).

COCOMO 81 was not designed for evolutionary software processes. There is no reference to the evolutionary prototyping or to the spiral model in the book (Boehm, 1981). The reason is because the spiral model was introduced later (Boehm, 1988). Moreover, in the recent book Boehm states that "COCOMO 81 did not plan for evolution", it was "built on the 1970's waterfall process framework" (Boehm et al, 2000 p. 3, 4). The Intermediate and Detailed COCOMO were designed "for cost estimation in the more detailed stages of software product definition." (Boehm, 1981 p. 114). The Intermediate and Detailed models require the introduction of subjective cost estimators. The Detailed Model was designed to for cost estimation in base of "phase distribution effort." But the word "phase" here refers to the phases in the waterfall not to the cycles in evolutionary prototyping.

COCOMO Intermediate and Detailed can be used for adaptations of existing software, but always requiring an input in terms of lines of code. The estimation of the adaptation size require to know:

- The number of delivered lines of code adapted from the existing software to form the new product.
- The percentage of design modified.
- The percentage of code modified.
- The percentage of effort required to integrate the adapted software.

The *adaptation* should not be confounded with a *build* in the evolutionary software process. The adaptation is related to the maintenance. It is a construction of a new product based on a previous developed software following the waterfall model.

COCOMO II can be configured for different software processes including the evolutionary ones. However, it requires an estimation of the size of the product at each evolutionary cycle (Boehm et al, 2000).

2. Putnam

In the 50's, Peter Norden from IBM developed a manpower model. He used the following curve of the Weibull distribution family, named after the 19th century physicist Lord Rayleigh:

$y = K (1 - \exp(-at^2))$, and its first derivative
$y' = 2 K a t \exp(-at^2)$, where
y = cumulative percentage of total effort
y' = manpower rate in terms of people per unit of time
K = effort in man-unit of time
t = development time
a = a constant governing the time to manpower peek.

Putnam, an alumnus of the Naval Postgraduate School, introduced in the 70's a model applying the concepts developed before by Norden at the IBM development laboratory of Poughkeepsie. This model is supported by a commercial tool named SLIM (Software Life Cycle Management). The use of the Rayleigh curve as a reasonably good fit for the manpower distribution has been proved by Norden, (Putnam, 1980) and (Boehm, 1981). Putnam observed that a strong correlation between lines of code and schedule, manpower and defects exists. He recognized differences in terms of development difficulties between real time systems and normal information systems (Putnam, 1980 and 1996). Putnam's model is based on the following assumptions (Londeix, 1987):

- A development project is a finite sequence of purposeful, temporally ordered activities, operating on an homogeneous set of problem elements, to meet a specified set of objectives.
- The number of problem elements is unknown but finite.
- Problems are detected, recognized and solved by applying effort.
- The occurrence of problem solving follows a Poisson process.
- The number of people working in the project is proportional to the number of problems to be resolve at that time.

The main equation of this model relates the size of the project in lines of code to the effort and the schedule:

$S = C_k K^{1/3} t_d^{4/3}$, where

S = number of delivered source instructions

K = life-cycle effort in man-years

t_d = development time in years

C_k = "technology constant" that requires fine-tuning

The required development effort (DE) is estimated as 40% of the life-cycle effort. That is:

$$DE = 0.4\ K = 0.4\ (S/C_k)^3\ (1/t_d^4)$$

One difficulty of the approach, as with COCOMO, is the need of knowing the number of lines of code at the beginning of the project. Putnam suggests using the Delphi method to estimate S.

Let a = minimum size estimation,
b = most likely size,
c = maximum size estimation.
The estimator of the expected size, E(S) = (a + 4b + c) / 6.
And the estimator of the standard deviation is s = (c - a) / 6.

Another difficulty is to estimate the technology constant C_k. Putnam suggests deriving it from previous projects. That is, analyzing post-mortem projects with known S, K and t_d it is possible to derive the value of C_k. This approach introduces two constraints:

(1) To apply the model, available historic data is required.

(2) The development process must be repeatable, that is at least CMM level 2.

(Boehm, 1981) states that this method is not good for projects employing incremental development, but this comment could be a little biased. Nevertheless, changes in requirements lead to a new estimation. According to Putnam, the method is not precise for small projects with development time of two years or less. This seems to be because of a more rectangular manpower pattern observed in small projects. The method has been verified with more than 4,000 projects. Conte also observed that the model works "reasonably well" on very large systems but overestimates the effort on medium and small ones (Conte, 1986). Other criticisms of the same authors point to exaggeration of the effects of time compression, excessive weight on the size, and excessive sensibility to changes of the technological constant.

During this research an experiment was conducted to compare Putnam's model with COCOMO 81. The experiment consisted in comparing the estimates of 100 projects with sizes from 10KLOC to 1MLOC using Basic COCOMO for organic, semidetached and embedded systems with Putnam estimation. To avoid problems of tuning, the effort in Putnam used the average of the development times of COCOMO. Similarly, the time in Putnam was calculated using the average of COCOMO efforts. In both cases a constant of technology = 10100 as suggested in (Boehm, 1981) was used. The following graphs show the findings:

(1) In terms of effort, Putnam's model is almost the average of embedded and semidetached COCOMO (Fig. 2.3).

(2) In terms of development time, the models are quite similar, Putnam's estimation being more optimistic (Fig. 2.4).

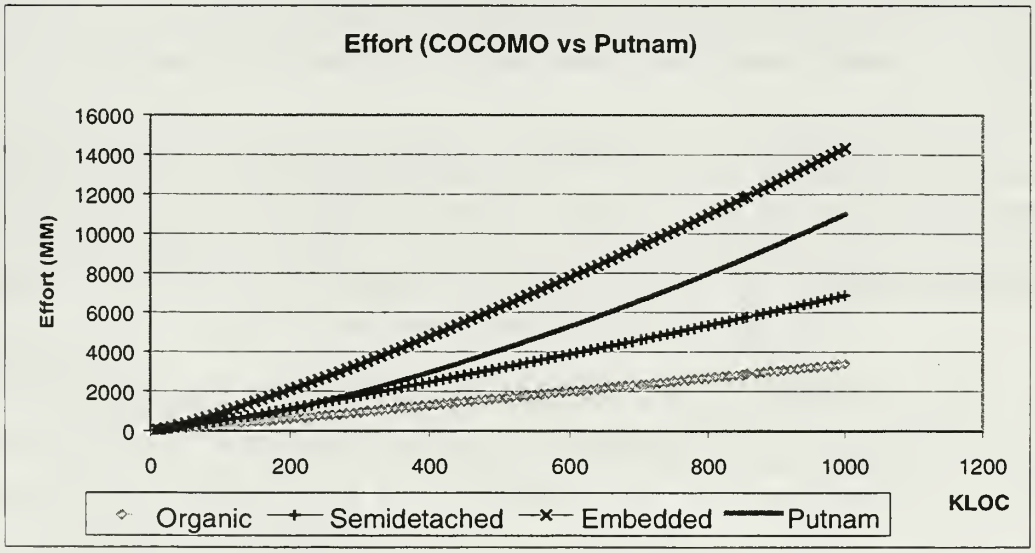


Figure 2.3: Effort Estimated Using COCOMO and Putnam Models

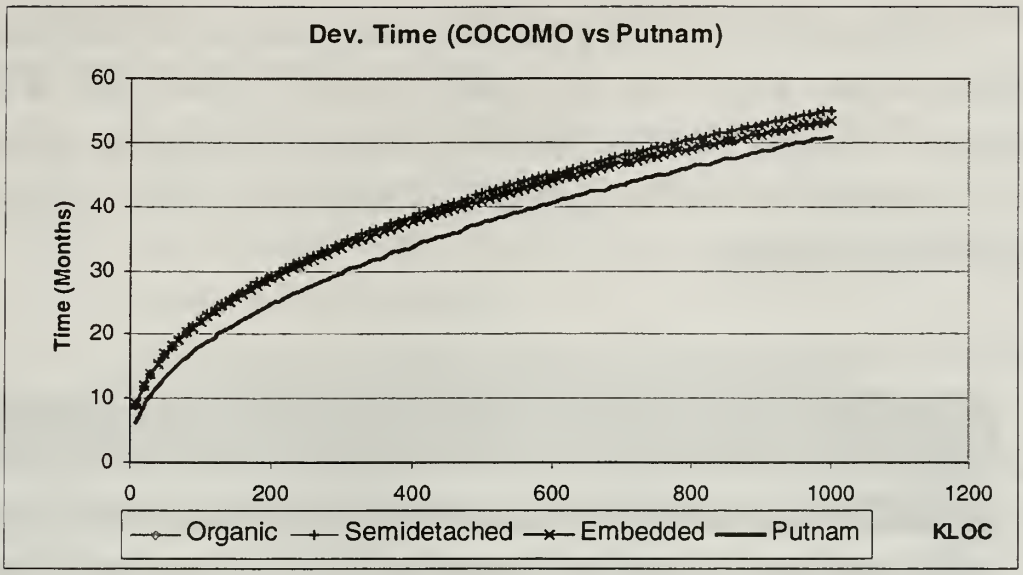


Figure 2.4: Development Time Estimated Using COCOMO and Putnam Models

3. Function Points

Functional complexity has been studied for years because it correlates highly with effort and risk. The traditional functional complexity metric has been introduced by (Albrecht, 1979 and 1983). Note that functional complexity includes to notions of complexity. First, there is the notion of relational complexity describing the mechanistic view of the system. This notion can be objectively measured. Second, there is a rational notion of complexity that is subjective and depends on cognitive limitations of the observer. Function Points had an enormous success because:

(1) It is an early metric. It can be calculated after the preliminary analysis of the system.

(2) It is easy to calculate. There are only five input parameters to compute and fourteen fine-tuning adjustments, but the whole process can be done manually.

(3) It is the first metric that related complexity to number of lines of code.

The procedure for calculating Function Points is quite simple. Count the number of inputs, outputs, queries, files, and system's interfaces required. Each of the five parameters is classified into simple, medium or complex. Depending on the parameter and its complexity, the count is multiplied by a weight factor. Table 2.6 presents the template for the calculation.

Table 2.6: Function Points Calculation (Albrecht, 1983)

	Simple	*Weight*	*Medium*	*Weight*	*Complex*	*Weight*	*Total*
Inputs	(	* 3) +	(	* 4) +	(	* 6) =	
Outputs	(	* 4) +	(	* 5) +	(	* 7) =	
Queries	(	* 3) +	(	* 4) +	(	* 6) =	
Files	(	* 7) +	(	* 10) +	(	* 15) =	
Interfaces	(	* 5) +	(	* 7) +	(	* 10) =	
					NAFP	=	Σ

The result of the total is called Non Adjusted Function Points. Fourteen adjustment factors, whose values are in the range of zero to five, describing the environment are added. Finally the Function Points are calculated by the formula:

$$FP = NAFP * (0.65 + 0.01 * \Sigma F_i)$$

where NAFP is the non adjusted Function points

F_i is each of the fourteen adjustment factors

Despite its attractive approach, Function Points has many weaknesses. First of all, the metric was derived from a study of MIS projects in the seventies. Today, there are many issues that are not considered by the metric and that are contributors to complexity. For instance, recursive functions, reuse, inheritance, communication by messages and polymorphism are not considered by the metric. The languages have evolved also and differ a lot from the COBOL of the seventies. Finally, programming styles have suffered a dramatic change that is not reflected in the metric.

(Kemerer, 1993) reported some weaknesses of the metric. Similar results have been reported by (Kitchenham, 1993 and 1997). The main issue is that Function Points is

a not well-formed metric because there is a correlation between their constituent elements. In her conclusions she stated that:

(1) The individual function point elements were not independent.

(2) Not all the function point elements were related to effort.

(3) An effort prediction metric based on inputs and outputs was just as good a predictor as Function Points.

(4) An effort prediction metric based on the number of files and the number of outputs was only slightly worse that Function Points.

(5) To get good estimates estimation methods and models based on the organization's performance, working practices, and software experience were required.

(6) Uncertainty and risk cannot be managed effectively at the individual project level. However, they can be managed in the organization context. If a single project had to be assured against all possible risks and uncertainty, its cost would be prohibitive. The sources for estimate uncertainty are the measurement error caused by model limitations and accuracy, the use of erroneous assumptions, and the use of the model outside its domain.

Even if evidence of defects in the metric existed, nobody introduced a better alternative. So, Function Points remained as the most common prediction metric for many years. More recently, some extensions to Function Points have been introduced, such as "feature points" and "Boeing's 3-F function points," addressing the effort estimation for embedded systems.

4. Conclusions about COCOMO, Putnam and Function Points

All these methodologies have some weaknesses with respect to software evolution. First, the need of a size estimate as an input parameter limited the applicability of COCOMO and Putnam methods. Second, the characteristics counted on function

points are quite different from the specification attributes. Third, the criticisms introduced by (Kemerer, 1993) and (Kitchenham, 1993 and 1997) suggested that despite the correlation observed between complexity and size, other metrics could be more accurate, and this opened opportunities for new research.

E. SOFTWARE RELIABILITY

Software reliability engineering is based on a solid body of knowledge that includes operational profiles, random process software reliability models, statistical estimation and sequential sampling theory (Musa, 1995). This section describes the reliability models and classifies them according to the characteristics of their probabilistic assumptions. Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment (ANSI, 1991). Software reliability is related to quality. ISO 9000-3 specifies the measurement of field failures as the minimal required quality metric. It includes:

a) Software reliability measurements, which includes estimation and prediction models.

b) Metrics and attributes of product design, development process, system architecture and their relation with reliability.

c) The application of the knowledge in specifying and guiding software development, system architecture, testing, acquisition, use and maintenance. However, the reliability approach studies the post implementation behavior of the software. At that stage, the product is already constructed. Hence, changes in the product are impossible or very expensive. So the "knowledge" arrives too late to be useful in the present project, but can be applied to future developments. (Lyu, 1995).

Donnelly, Everett, Musa and Wilson stated that the practice of software reliability provides a means to "predict, estimate, and measure the rate of failure occurrences in software and firmware" (Lyu, 1095 pp.219). Reliability can only be reached by following

a rigorous development process. During the product concept development, it is necessary to determine the functional profile, define and classify the failures, and identify the customer's reliability needs. Determining the functional profile means to specify the tasks to be performed and the environmental factors that could influence the process. Quality Function Deployment (QFD) is a useful methodology to apply. The definition and classification of failures must be done from the point of view of the user and represents a trade-off. Too many classes require excessive effort in collecting and analyzing the metrics. Few classes can provide too vague information. Finally, identifying the user reliability needs at a high level is required. A reasonable way to do this is to assess the reliability capabilities of similar products.

During the requirements phase, reliability objectives need to be refined and specified, including customer satisfaction, performance, and trade-offs between reliability and other factors, such as cost, delivery time and functionality. The reliability requirements have strong influence in the architecture and in the future evolution steps of the development process.

During the design phase, one must analyze the reliability of the components in order to determine the overall reliability of the system. This design phase should be driven by reliability objectives after the identification of risky areas. For instance, the use of redundant software elements could be required.

During the implementation, one should identify the critical areas and different techniques that should be applied to reduce risks. Such techniques include tight development standards and methodology, modularity, reuse, development in evolutionary steps, inspections and reviews, and software configuration management. Another important consideration in this phase is the reliability assessment for components acquired or developed by outsourcing. This assessment must be done as soon as possible.

Part of the testing (unit test and integration) is conducted during the successive evolutionary steps, but in the system test many reliability issues could appear. At that time, the system is complete, and we can have a complete picture of its reliability prior to the delivery to the customer. To conduct the system test, one must determine various operational profiles according to the user's point of view. The purpose is to locate stress points where the reliability requirements are not reached or could potentially not be reached. This is a particularly intensive data collection activity that requires automated tools. Testing could continue in beta test sites where the reliability objectives are certified.

Finally, when the product is delivered to the customer, reliability should be monitored to assess the success of the project, to measure the quality of the process including the testing scenarios, and to have an indicator of customer satisfaction. The last is a critical success factor that needs to be tracked also by surveys and meetings, in order to be sure that any symptom of dissatisfaction was immediately revealed. Software reliability techniques are another tool available that can improve the software development process. Unfortunately, the models available require that the product is almost complete in order to predict its behavior. In the following sections we will describe various software reliability models classified according to the following scheme:

a) Exponential failure time models
b) Weibull and Gamma failure time models
c) Infinite failures models
d) Bayesian models
e) Models for early stages

1. Exponential Failure Time Models

a. Jelinski-Moranda Model

Jelinski and Moranda introduced this model when they were working for McDonnell Douglas. The elapsed time between failures has an exponential distribution

with a parameter that is proportional to the number of remaining faults in the software. Although this model has been replaced, it was important for setting the framework for other work in modeling. The assumptions for this model are the following:

- The rate of fault detection is proportional to the current fault content of the software.
- The fault detection rate remains constant in each interval between faults.
- The correction of a fault is instantaneous and does not introduce new faults.
- The operation of the software is similar to the conditions in which the prediction of reliability is done.
- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

Model form

The time between failures $X_i = t_i - t_{i-1}$ for $i = 1,\ldots n$

$\therefore$ X_i are independent random variables with exponential distribution, with

Mean failure $= \mu(t) = N / (1 - \exp(-\phi t))$

Failure intensity function $= \lambda(t) = N \phi \exp(-\phi t)$

Where N = total number of faults in the software

ϕ = proportionality constant.

b. Schneidewind Model

This model introduced by Dr. Norman Schneidewind in (Schneidewind, 1975) is based on the idea that the current fault rate is a better estimator of the future behavior than the observed rates in the distant past because the failure rates can change over time. So, the model weights the observations differently, according to the analyst's point of view. The main idea introduced by Schneidewind was to monitor the occurrence of software errors as a predictor for future cumulative detected and corrected errors. These estimations are useful in order to:

- Identify the trade-off function between error reduction and cost of error reduction.
- Provide a quantitative basis for accepting or rejecting software during functional testing.
- Provide a quantitative basis for deciding whether additional testing is warranted based on the cost of error removal.

The model is based on non-homogeneous Poisson processes. A non-homogeneous Poisson process is a Poisson process in which the mean is not a constant, but a random variable. This model has been used extensively on IBM's Flight Control software for the Space Shuttle with great success. It is one of the four selected models by the AIAA's Recommended Practice for Software Reliability (AIAA, 1993) and considered one of the most accurate available (Lyu, 1995). Schneidewind proposed three forms of the model:

Model 1: Uses all fault counts of the n periods, reflecting the view of equal importance.

Model 2: Ignores the fault counts of the first s-1 periods. This reflects the view that the early time period contribution in predicting future behavior is insignificant. This is very useful to discard the confounding effect of a learning curve.

Model 3: Uses the cumulative-fault counts from intervals 1 to s-1 as the first data point, and the individual counts for periods s to n, as additional data points. This view is an intermediate between the other two.

The assumptions for the models are the following:

- The cumulative number of failures by time t follows a Poisson process with mean $\mu(t)$ such that the expected number of fault occurrences for any time period is proportional to the expected number of undetected faults at that time.
- The number of faults is finite.
- The failure intensity function decreases exponentially with time. The failure intensity function $\lambda(t) = \alpha \exp(-\beta t)$ for some α, β constants.

- The number of faults (fi) detected on each interval i are independent.
- The fault correction rate is proportional to the number of faults to be corrected.
- All the intervals have the same length.
- The operation of the software is similar to the conditions in which the prediction of reliability is done.
- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

Concerning error detection, Schneidewind's model has the following assumptions:

- The observation of the process is in discrete intervals of time.
- The number of errors in each time interval is independent of the number of errors in any other interval.
- The pdf in each interval has the same distribution but a different mean.
- The mean of number of detected errors decreases from interval to interval as a result of the correction process.
- The rate of error detection in an interval is proportional to the number of errors in the interval.

Because of (2) and (3) this model is a non-homogeneous Poisson process with exponential decaying intensity function as stated in (4). The following equations summarize the detection model:

x_i = actual number of errors during interval i

m_i = estimate number of errors during interval i

Decaying intensity function $d(i) = \alpha \exp(-\beta i)$ for $\alpha, \beta > 0$

Cumulative mean number of errors $D(i) = (\alpha/\beta) \exp(-\beta i)$

Estimate number of errors during interval i

$$m_i = (\alpha/\beta) \{\exp(-\beta(i - 1)) - \exp(-\beta i)\}$$

Time estimated to detect D cumulative errors

$$i_d = \{\log (\alpha/(\alpha - \beta D))\} / \beta$$

Time estimated for the detection rate to reach the value d

$$i'_d = (\log (\alpha/d)) / \beta$$

Delay correction error. This is the difference in time between detection and correction of errors.

$$C(i) = D(i - \Delta i) = (\alpha/\beta) \{1 - \exp(-\beta(i - \Delta i))\}$$

Time estimated to correct a cumulative number of errors

$$i_c = \Delta i + \{\log (\alpha/(\alpha - \beta C))\} / \beta$$

Correction rate of errors $c(i) = \alpha \exp(-\beta(i - \Delta i))$

Time estimated to reach a correction rate c

$$i'_c = \Delta i + (\log (\alpha/c)) / \beta$$

Difference between detected and corrected errors

$$R(i) = D(i) - C(i) = (\alpha/\beta) (\exp(-\beta i))(\exp(\beta \Delta i) - 1) \quad \text{for } i \geq \Delta i$$

Unlike hardware, which deteriorates with time, software ideally should improve with time. Theoretically as time goes by more errors are discovered, so the residual number of errors decreases. However, this is not always the case. Two factors contribute to reproduce errors in a software system. First, we need to consider that error correction is error prone. New errors could be introduced as a consequence of debugging. Second, and more important, software require maintenance and these major modifications or extensions are source of new errors. Consequently, the time series of error counts will not necessarily be monotonically decreasing. Our concern about the model is that the assumption of correction without introducing new errors seems to be optimistic.

c. Goel-Okumoto Model

This model uses the number of faults per unit of time as independent Poisson random variables. It was introduced in 1979 by Goel and Okumoto (Goel & Okumoto, 1979) and is the source for other models, such as the S-shaped model. The assumptions for this model are the following:

- The cumulative number of failures at time t follows a Poisson process with mean $\mu(t)$. This mean function is such that the expected number of fault

occurrences for any time interval (t, t+Δt) is proportional to the expected number of undetected faults at time t. It is also assumed that the total number of faults is finite.

- The number of faults detected in each time interval is independent for all time.
- The operation of the software is similar to the conditions in which the prediction of reliability is done.
- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

Model form

Mean = $\mu(t) = N (1 - \exp(-b\ t))$ for some constants b, $N > 0$

Failure intensity function = $\lambda(t) = N\ b \exp(-b\ t)$

A variation of this model permits one to determine an optimal release time for a software system. This adaptation uses the time of fault occurrences instead of the fault counts. Given a desired reliability R for a specified operational time O, then the required amount that the software must be observed is:

$$T = (1/b) (\ln(a(1 - \exp(-b\ O)) - (\ln(\ln(1/R))))$$

d. Musa's Model

The rationale behind this model is that the execution time is more reflective of the actual stress applied to software than the calendar time. Dr. John Musa of AT&T Bell Laboratories (Musa, 1975) introduced the model in 1975. The assumptions of the model are the following:

- The cumulative number of failures by time t (M(t)), follows a Poisson process with mean $\mu(t) = \beta_o (1 - \exp(-\beta_1 t))$, where $\beta_o, \beta_1 > 0$. The expected number of failures in any period is proportional to the expected number of

undetected failures at that time. The total number of faults that will be detected when time $\rightarrow \infty$ is β_0.

- The execution times between failures are exponentially distributed.
- The personnel resources remain constant over the period of time the system is observed.
- The system assumes a relationship between MTTF and resource expenditures.
- Testing personnel can be fully utilized and computer utilization is constant.
- Fault-correction personnel are assigned randomly to serve a fault queue. Fault correction is assumed to be a Poisson process.
- The operation of the software is similar to the conditions in which the prediction of reliability is done.
- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

e. Hyperexponential Model

Ohba in 84 (Ohba, 1984) introduced the first hyperexponential model. The main idea of this model, which has many variations, is that a system has different classes of software. The software has an exponential failure rate, but each class or section has different rates reflecting its nature (differences in complexity, differences between developers, or differences in programming languages). When there are only two classes (e.g. old software versus new software, or easy to test versus difficult to test, etc.), this model is called a modified exponential software-reliability-growth model. The model has the following assumptions:

- The software is composed by K sections (or classes of code) so that within each class:
  - The rate of fault detection is proportional to the current fault content.
  - Fault detection is constant over the intervals between faults.

- Each fault is corrected instantaneously without introducing new faults.
- The software as a whole has a cumulative number of failures by time t (M(t)), that follows a Poisson process.
- The operation of the software is similar to the conditions in which the prediction of reliability is done.
- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

2. Weibull and Gamma Failure Time Models

a. Weibull Model

This model assumes that the fault distribution has a Weibull distribution. Many hardware failure models are modeled with this distribution (Lyu, 1995). The main characteristic of the distribution is that its parameters permit great flexibility, adapting the model to increasing, decreasing or constant failure rates. The assumptions of the model are the following:

- There is a fixed number of faults at the beginning of the experiment.
- Each fault has a time to failure (Ta) distributed as a Weibull distribution with parameters α, β:

  $Ta = \alpha\beta t^{\alpha-1} \exp(-\beta t^{\alpha})$, with $\alpha, \beta > 0$ and $t \geq 0$.
- The number of faults detected in each time interval is independent.
- The operation of the software is similar to the conditions in which the prediction of reliability is done.
- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

b. S-shaped Reliability Growth Model

The per-fault failure distribution of the S-shaped model is Gamma (Lyu, 1995). The number of failures per period of time follows a Poisson process. This model

assumes a finite number of failures. The S-shape growth curve describe the testing process with an initial learning curve at the beginning, then a growth, and then a level at which errors are more difficult to detect. The S-shaped model has the following assumptions:

- The cumulative number of failures by time t, follows a Poisson process with mean = $\mu(t) = \alpha(1 - (1 + \beta t)e^{-\beta t})$ for $\alpha, \beta > 0$. At the limit, when $t \rightarrow \infty$, $\mu(t) = \alpha < \infty$.
- The time between failures of the (i - 1)st and the ith depends on the time to failure of the (i - 1)st.
- When a failure occurs, the fault is immediately removed without introducing new faults.
- The operation of the software is similar to the conditions in which the prediction of reliability is done.
- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

3. Infinite Failure Models

a. Duane's Model

This model was originally proposed for hardware reliability (Lyu, 1995). This is a non-homogeneous Poisson process in which the failure intensity function has the same form as the hazard rate of the Weibull distribution. This model as been referred to as "the power model." The assumptions of the model are the following:

- The cumulative number of failures by time t, follows a Poisson process with mean = $\mu(t) = \alpha t^{\beta}$ for $\alpha, \beta > 0$ (If $\beta = 1$, then we have the homogeneous Poisson process).
- The operation of the software is similar to the conditions in which the prediction of reliability is done.

- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

b. Geometric Model

The geometric model (Lyu, 1995) is a variation of the Jelinski-Moranda Model. Here the time between failures is assumed to be exponential distributed with mean decreasing geometrically. The geometric decay reflects the smaller impact of the later-occurring faults. The assumptions of the model are the following:

- The fault detection rate is a geometric progression, and it is constant between fault detections.
- There are an infinite number of total faults in the system.
- The time between detection follows an exponential distribution.
- The operation of the software is similar to the conditions in which the prediction of reliability is done.
- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

c. Musa-Okumoto Logarithmic Model

The Musa-Okumoto logarithmic Poisson Model (Lyu, 1995) is another example of nonhomogeneous Poisson processes with an exponential decrease. The decrease reflects the view that the earlier discovery of failures has a greater impact on reducing the failure intensity function than those encountered later. The assumptions of the model are the following:

- The failure intensity decreases exponentially with the expected number of failures experienced.
- The cumulative number of failures follows a Poisson process.
- The operation of the software is similar to the conditions in which the prediction of reliability is done.

- Within each severity class, the probability of finding a fault is the same.
- The failures are independent.

4. Bayesian Models

The Bayesian approach introduces a subjective viewpoint considering that if no failures occur while the software is observed, then the reliability should increase, reflecting the growth in user's confidence. The reliability is considered a reflection of the number of faults discovered and the time between failures.

The Bayesian approach also reflects the viewpoint that different faults have different impacts on the reliability of the system, and the number of faults is not as important as their impact. If many faults exist but they seldom appear, then by using the Bayesian approach the system is relatively reliable. The mean time to failure is therefore a very important metric in this framework.

a. Littlewood-Verrall Reliability Growth Model

This model tries to account for the fault generation during the fault correction process. This approach is very realistic because the software could become less reliable than before. Because of the uncertainty, each new version can be better or worse in terms of reliability. The distribution of failure times is random. The assumptions of this model are the following (Lyu, 1995):

- Successive execution times between failures are assumed to be independent exponential random variables with parameter ξ_i, i = 1,..., n.
- The ξ_i's form a sequence of independent random variables, each with gamma distribution.
- The software is operated in the specified or normal way.

b. Other Bayesian models

Other Bayesian models include variations of several models, such as Jelinski-Moranda, Jewell, Littlewood and Sofer, Kyparisi-Singpurwalla, Lyu, Becker-Camarinopoulos, and Thompson-Chelson (Lyu, 1995).

5. Software Reliability Prediction in Early Stages

The prediction of software reliability during testing is useful to understand the future behavior of the software in the operation phase. However, this benefit is limited because it is too late to incorporate changes without incurring high costs and schedule overruns. If a prediction could be made in the early phases, where changes can be introduced without excessive costs and without a serious impact on the schedule, then a drastic improvement could be reached. Moreover, this prediction could be a great advance in software engineering. Few models have been addressed to predict software reliability during the early phases of the project.

a. Phase-Based Model

Gaffney and Davis in (Gaffney, 1988) introduced the Phase-based model using statistics obtained during reviews requirements, design and implementation. The model is based on the following assumptions:

- The staffing level of the development effort is directly proportional to the number of faults discovered during each phase.
- The fault discovery curve is monomodal.
- A good estimation of code size exists.

The number of discovered faults per line of code from phase t-1 to phase t is given by:

$$\Delta Vt = E\,(\exp(-B(t-1)^2) - \exp(-Bt^2))$$

where E = the total lifetime fault rate is expressed in terms of KLOC

t = the discovering index associated with the phase (requirements: t = 1, design: t = 2, implementation: t = 3, unit test: t = 4, software integration: t = 5, system test: t = 6, acceptance test: t = 7).

B = a defect discovery constant.

A fourth assumption implicit in the model is that the development process follows the waterfall life cycle. This is an important restriction for adapting the model to the new evolution development processes.

b. Agresti-Evanco Ada Model

Agresti and Evanco introduced in (Agresti, 1992) a prediction model that addresses the particularities of Ada language. The model is a multivariate linear regression with dependant variable the log of the default density. Among the independent variables are architectural complexity of design, volatility, reuse and functional complexity. This model seems to have an interesting level of accuracy. According to (Lyu, 1995), 63 to 74% of the variation can be explained with this model. However, the applicability to Ada language projects restricts the model considerably.

c. Rome Lab Model

The US Air Force's Rome Laboratory introduced this model in 1992 (Rome Lab, 1992) to predict the initial fault density as a function of the following factors:

- Application type, differentiating projects in three categories: real time, scientific and managerial.
- Development environment, considering the differences in tools and methodologies according to three categories: organic, semidetached and embedded.
- Requirements and Design metrics: anomaly management, tracebility, and existence of software quality assurance in the process.

- Software implementation metrics: development language level, size, modularity, reuse, complexity, and the existence of review standards.

6. Conclusions about Software Reliability Models

The main contribution of this set of research is the emphasis on solid statistical foundations to assess reliability. Some of the distributions used, such as non-homogeneous Poisson processes and Weibull, are interesting to model life cycles. The caveat of this approach is that the conclusions from the use of those models arrive too late in the life cycle to provide effective support from the engineering point of view.

F. MODERN PROJECT MANAGEMENT TECHNQUES: VitéProject

1. VitéProject

VitéProject is a modeling and simulation tool that integrates the organizational work of projects explicating the interdependencies between tasks and roles not only from the point of view of producer-consumer, such as in CPM or Pert, but also communication and rework dependencies. VitéProject is the commercial version of VDT (Virtual Design Tool), a research based on contingency theory directed by Dr. Raymond Levitt at Stanford (Jin, 1996). CPM models are sequential interdependencies through explicit representation of precedence relationships between activities. This simplified vision of the project cannot address the dynamics created by reciprocal requirements of information in concurrent activities, exception management, and the impacts of actor interactions. This issue is addressed by VDT. The original model of VDT was based on the following observations about collaborative, multidisciplinary work in large complex projects:

- Organizational tasks in the project can be divided into two categories: production work that directly adds value to the product, and coordination work that facilitates the previous one.

- Contingency theory provides qualitative insights about the extent of coordination work, but did not provide information about how to address the bottleneck problems created by coordination.

The model integrates the micro-level description of the entities that perform work and process information called "actors." Actors can be individuals or small teams working as a unique and cohesive unit where individuals are not differentiated. Actors have two basic behaviors: *attention allocation* and *information processing*. As a consequence of such behaviors, actors perform *production* and *coordination*. The model is based on the following assumptions:

- Actor allocation assumption: Each actor has one input buffer where all the incoming information and requests for production or coordination work arrive. The input buffer is a queue that supports different policies: priorities, FIFO and random. Each actor also has an output buffer to place its accomplished work.
- Actor capacity allocation assumption: An actor has a certain information-processing capacity determined by its skill type, skill level, and allocable time. An information processing work can be processed and completed if the actor allocates sufficient capacity to the job. This assumption implies: a) information processing requires not only attention but also takes time; b) the information content of a work is related to the skills; c) the volume of a work is related to the time; d) actors have limited capacity to allocate.
- Actors cannot allocate 100% of their capacity to work because they are interrupted by: a) information requests from other actors; b) decision-making to solve exceptions produced by subordinate actors; c) meetings; and d) processing noise, that is all other interruptions created outside the project that impact the actor.

The organization structure is modeled through simulation. The organization variables, such as control structure, communication structure, formalization and matrix

strength, influence the actor's micro level actions, and consequently an organization's emergent performance appears.

In a Vité project, an activity is any work that consumes time and may generate communications or exceptions. Each activity has a set of properties that include: name, description, work (time), a number of sub-activities, priority, skill required, and QFD analysis measures (requirement complexity, solution complexity, and uncertainty) (Levitt, 1999). For each activity in the project, VitéProject requires an estimation of its complexity in terms of its effort or duration. This is expressed in FTE (full time equivalent). One person working full-time is one FTE (Levitt, 1999). The work required for the activity depends on its complexity, and is estimated as the total time (including time of all necessary workers) required to do effective work on the activity. This does not include ineffective work, such as rework, waiting, or coordination. This number is sometimes referred to as the "work volume" or the "effort hours" for the activity. The activity duration is computed as: duration = (work volume)/(FTE assigned).

Complementarily, Vité provides adjustments to refine the complexity of each activity, and to express the difficulty of the elucidation of the requirements. VitéProject provides three parameters to adjust the complexity and the uncertainty of the requirements (Levitt, 1999)[5]:

1. *Solution Complexity* refers to the extent to which an activity's solution is affected by other activities.
2. *Requirements Complexity* models the number and difficulty of the functional requirements that need to be satisfied to complete the activity.
3. *Uncertainty* represents the extent to which information needed to complete an activity is unavailable at the time the activity starts. The missing information could be the output of a concurrent activity, the missing information about a client requirement, or the missing information about an unknown state of nature.

[5] The details of the configuration of VitéProject are presented on Chapter V, Section C.6 and Table 5.1.

In conclusion, *complexity* is mainly expressed in terms of *effort* (FTE of each activity in the work-breakdown structure) and secondarily by the parameter *solution complexity*. *Requirements volatility* is mainly expressed by the parameters *requirements complexity* and *uncertainty*. This is the approach used in this dissertation.

2. Validation of VitéProject[6]

The Virtual Design Team (VDT) research was initiated in the late 1980s at CIFE[7] with the goal of developing a new micro-organization theory and embedding it in software tools that could be used to design organizations in the same way that engineers design bridges, semiconductors or airplanes, by modeling, analyzing and evaluating multiple virtual prototypes of the system to be designed on a computer. The research concluded that attempts to model organizations computationally could benefit greatly from the use of non-numerical or symbolic representation and reasoning techniques emerging from computer science research on artificial intelligence.

VDT theory and analysis tools for project organizations have enabled true "organizational engineering" of project teams with congruent goals and routine-albeit complex and fast-paced-design or product development work. Dr. Levitt says that "our intention was always to start with the 'organizational information flow physics' and then progressively add elements of 'organizational chemistry' to the modeling framework. This would allow us to move out of the easy corner of the organizational space and address a wider range of tasks and organizations. It is useful at this point to position our completed and ongoing versions of VDT in the space of organizations and modeling issues." (Levitt, 2000).

---

[6] The author thanks Dr. Raymond Levitt (Stanford University), Carlos Rivero (Stanford University), and Raymond Buettner (NPS) for their support.

[7] CIFE is the Center for Integrated Facility Engineering at Stanford University.

The project has been developed in successive layers of research. The VDT-2 framework, which is the base for VitéProject, has been fully validated at different levels:

- micro-level analysis, using toy problems
- meso-level analysis, using toy problems, and experiments
- macro-level analysis, by testing for authenticity, reproducibility, generalizability, and prospective.

The validation strategy included different validation techniques, such as (Thomsen et al., 1999):

- Toy problems: used to analyze micro behaviors.
- Intellective simulations: the results from idealized simulations with extreme values were compared to the outcomes predicted by organizational theory.
- Reasoning and representation: validating a researcher's ability to model and simulate a real organization using the system.
- Authenticity: validation of the representation of a real organization.
- Generalizability: following the evolution of real projects from the industry for a period of three years.
- Retrospective validation: duplicating past performance using a simulation model and calibrating the model as needed to reproduce previous experiences.
- "Gedanken"[8] validation: using "what-if" questions and observing the predictions against experts' opinions.
- Natural history validation: by comparing the predictions for a simulated organization to the observations on the real organization.
- On the field: VitéProject has been used with success on diverse industries, such as shipbuilding (Det Norske Veritas), petrochemical (Dow Chemical, ePM, Shell), construction engineering (Macomber Co.), airlines (American), pharmaceutical and biotechnology (Pharsight), utilities (PG&E), manufacture (Procter & Gamble), machinery (John Deere), consulting (Integrated Project Systems, Vité Services Group, American Century), electronics and technology (Agilent Technologies, Dell, Hewlett Packard, Silicon Graphics, Applied

Materials), aerospace (Lockheed-Martin), and of course software engineering (Master Systems, Hewlett Packard). [9]

Cook and Campbell introduced a validation framework for experiments (Cook & Campbell, 1976). They extended a previous work from (Campbell & Stanley, 1976) introducing four kind of validation:

- Statistical validity, which refers to conclusions derived from statistical evidence.
- Internal validity, which refers to conclusions derived from demonstrated cause effect relationships.
- Construct validity, which refers to the inexistence of confounding factors affecting the logical chains of causation.
- External validity, which relates to the correspondence between samples, the populations they represent, and the populations to which generalization is required.

Table 2.7 compares the validation strategy followed in VitéProject (Thomsen et al., 1999) to the framework introduced by Cook and Campbell, and shows that the validation of Vité is fully compliant with this framework.

---

[8] The German verb, meaning "to think."

[9] More references about the use of Vité on software were introduced in a paper by (Rifkin, 2000) and (Nogueira et al., 2000c) both papers were presented at the International Conference on Software Engineering (Limerick, Ireland, June 2000).

Table 2.7: Comparison between VDT-Vité Validation Strategy (Thomsen et al., 1999) **and Cook-Campbell Framework** (Cook & Campbell, 1976)

VDT-Vité Strategy	*Statistical Validity*	*Internal Validity*	*Construct Validity*	*External Validity*
Toy problems validation	yes	yes	yes	
Intellective simulations Validation		yes	yes	
Reasoning and representation validation			yes	yes
Authenticity Validation			yes	yes
Generalizability validation				yes
Retrospective validation	yes			yes
"Gedanken" validation	yes	yes	yes	yes
Natural history validation	yes		yes	yes
Prospective validation	yes	yes	yes	yes

The VDT framework, which explicitly models information dependency and failure propagation between concurrent activities, has proven to be far more accurate than CPM/Pert models (Thomsen et al., 1999). Although CPM/Pert tools have not been designed specifically for software projects, they constitute a common practice for software engineering projects and have been applied with success for decades. Moreover, they are a recommended practice (Humphrey, 1980), (Pressman, 1992), (Sommerville, 1992). The reason is they have been designed on the basis of an abstract view of a project (a digraph), which is applicable to various types of projects. VitéProject has the same level of abstraction. The only difference is that unlike CPM/Pert, Vité explicitly represents the coordination among activities, the probabilities of failure, and the information exchange intensity. Vité/VDT projects have the classical time dependency between tasks, but they are also connected to an organization that processes information. This view is supported by (Marsch & Simon, 1958), (Simon, 1976), (Galbraith, 1977).

VDT is applicable to projects, in which:

- All activities in the project can be predefined.
- The organization is static, and all activities are pre-assigned to actors in the static organization.
- Exceptions to activities result in extra work volume for the predefined activities and are carried out by the pre-assigned actors.
- Actors are assumed to have congruent goals.

Such conditions are congruent with the breakdown structure of a software process. Moreover, VitéProject showed high accuracy in software projects.[10] VDT was designed to model organizations that deal with great amounts of information processing and coordination. Such characteristics are extremely relevant in software processes (Boehm, 1981). To improve the realism, the VitéProject parameters used in this dissertation were fine-tuned using Organizational Consultant (Burton & Obel, 1998) as described in Appendices A and B. Details of the validation process for Organizational Consultant are presented in Section G.4.

More details about the philosophy of validation for VDT can be found in (Thomsen et al. 1999). Successive doctoral dissertations at Stanford (Cohen, 1991) and (Christiansen, 1993) validated VDT retrospectively and concurrently against managers' predictions. In addition to the theses, all available as Stanford Ph.D. dissertations, (Kunz et al. 1998) reports the validation conducted at Lockheed. The validation process found that VDT provides an excellent first-order theory to employ as the basis for building models that can predict the flow of knowledge work through organizations. Those models can help managers diagnose and address information bottlenecks, delays, and quality problems arising from failed communication attempts. Appendix E presents a list of 74 books, papers in refereed journals, papers in conferences, research reports, and CIFE

[10] Master Systems reported that the typical error between predicted and actual project durations after the work-breakdown structure is available is 10%.

publications that explain the theoretical foundations and details about VDT, VitéProject and their validation. The value of the parameters used in the simulations are discussed in Chapter V, Section C.6.

3. Validation of Organizational Consultant[11]

Because the simulation model was created for general purposes, it is necessary to tailor it to reflect the desired organizational characteristics (Thomsen et al, 1999), (Burton & Obel, 1998). Since the parameterization of VitéProject used the expert system Organizational Consultant (OrgCon), insights about its validation are included in this section. OrgCon is an expert system designed to diagnose organizational problems and recommend organizational changes. One can determine the parameters for an organization given a description of its goals, environment, and internal characteristics. The tool was used to determine the organizational parameters for Vité.[12] The tool perfectly matches VitéProject because both groups of research have been interconnected and mutually collaborative. That is the reason for the choice of Vité and OrgCon in this dissertation. As stated by (Burton & Obel, 1998. pp xx): *"Ray Levitt at Stanford (the director of the VDT project) also used both the book and Organizational Consultant in his course. Ray's many comments helped improve the final version."* The terms used on Vité and OrgCon are the same, as well as the definitions provided in the manuals and help facilities. The research for the tool was supported by the Danish Social Science Research Council, Duke University, and Odense University.

The validation process of OrgCon relies on information obtained from cases, consultation with executives, dialogue with experts, and MBA courses. The validation process followed the literature trends in validation. The validation process for OrgCon followed the framework form (Cook & Campbell, 1976) discussed in the previous

---

[11] The author thanks Roxanne Zolin, a good friend from Stanford, who planned and managed the OrgCon project and who also was part of the VDT-Vité group for her help and support.

[12] The organizational characteristics for software development revealed by OrgCon are discussed at the end of Section H.1.

section, but it is also compliant with the O'Leary method. (O'Leary, 1988) developed a schema to validate an expert system based on six steps:

- analyze the knowledge base for accuracy
- analyze the knowledge base for completeness
- analyze the knowledge base weights
- test the inference engine
- analyze the decision quality
- analyze the condition-decision matches

The validation of OrgCon merged both Cook-Campbell and O'Leary approaches. The internal and statistical analysis corresponds to the three first steps of the O'Leary method. The external and construct validity correspond to the remaining three steps of the O'Leary method. The validation process started with the creation of a prototype that evolved from an initial set of rules derived from literature, increasing the knowledge base on each iteration. After several cycles, the prototype was reviewed. The next validation step was the development of a set of test cases. These test cases were tested with executives, experts and MBA students. Finally, observations and modifications did the fine-tuning. The domain of applicability for which OrgCon has been validated include among others, software development organizations,[13] including companies of different sizes (from ten to 3000 persons) (Baligh et al, 1994). Further details of the validation process can be found in (Baligh et al., 1994) and (Baligh et al., 1996).

[13] The author thanks Dr. Borge Obel (Odense University) and Dr. Richard Burton (Duke University) for their support. The following is part of the e-mail received from Dr. Obel regarding the validation of OrgCon: "We have validated OrgCon using a set of some 20 test cases as well as tested the program in a number of companies. Additionally, it has been used by several hundred students both on cases and real companies. These companies include *software development* companies. We have written a paper detailing the validation process."

G. ORGANIZATIONAL THEORY

This section introduces some foundations of organizational theory that support the research. Why review the organizational foundation if the research is about software engineering? First, software development requires teamwork, more specifically organized work. So one must understand the dynamics of organizations as artificial social entities that exist to achieve a specific purpose, in this case to develop software. Second, organizations are composed of individuals who accomplish diverse activities that require coordination and consequently an information exchange. Coordination and information exchange, despite their impact, have not been covered by the research in estimation models. Third, VitéProject was developed for general projects. In order to obtain a rigorous simulation, one must customize the tool according to the characteristics of software engineering.

1. Introduction

As software systems increased in complexity, software development evolved from a primitive art into software engineering. Methodologies and software tools were developed to help the development processes. Most of the present tendencies (DOD-STD-2167A, ISO-9001, SEI/CMM) try to standardize processes, emphasizing planning and structure (Humphrey, 1990). Some authors criticize those approaches stating that they underestimate the dynamics of the software development (Bach, 1994), (Abdel-Hamid, 1997). Others question that activities such as research and development are not addressed by TQM principles (Dooley et al., 1994). In the author's opinion, many of the problems on current software projects have organizational roots. This view is also supported by (van Genutchen, 1991)[14] and (Jones, 1994).[15] The typical software engineering process is a succession of decision problems trying to transform a set of fuzzy expectations into requirements, specifications, designs and finally code and documentation. The traditional

---

[14] Van Genuchten found that 45% of all the causes for delayed software are related to organizational issues.

[15] Capers Jones found that on military software developments the two more common threats are excessive paperwork (90% of the time) and low productivity (85% of the time).

purpose because it applied a method valid for well-defined and quasi-static scenarios. This hypothesis is far from the reality. Today, modern software processes (Boehm, 1988), (Luqi, 1989) are based on evolution and prototyping. These approaches recognize that software development presents an ill-defined decision problem, and they fail to assess the risk automatically. Software development projects present special characteristics that must be solved in order to improve the state of the art. These particularities affect the *strategic planning*, the *organizational structure*, and the *engineering applied to software*.

Computational organizational theory (COT) is the study of organizations using computational techniques. It views organizations as complex information processing systems composed of multiple distributed agents that exhibit organizational properties. These agents are assigned tasks, resources and responsibilities. Agents accomplish their tasks processing information and asking for collaboration from other agents (Prietula et al, 1998). Organizations are complex, dynamic, nonlinear adaptive and evolving systems. Agents (artificial and human) exhibit adaptive behaviors. For that reason the emergent organizational behaviors are difficult to predict by analytical methods. However, computational analysis could be a valuable tool for the study of these behaviors to understand concepts, or to determine the consistency of a theory, particularly concerning high-performance organizations.

High-performing organizations are characterized by their capacity to deal with rapidly unfolding events with high uncertainty and potentially catastrophic impacts (Kang et al., 1998). The need for studying the dynamics of teams has been demonstrated by incidents such as those that involved the USS Stark and the USS Vincennes. A team is a group whose members share a common goal and a common task. Teams differ from generic groups in their high differentiation and the interdependency of its members. Groups consist of homogeneous and interchangeable members with little or no interdependency. The software development process is based on the work of teams that act as problem solvers for one or more tasks.

There are some differences in solving problems between teams and individuals. Teams act in a distributed fashion and search for strategies more efficiently. Teams decompose the problem into subproblems that are analyzed simultaneously and then integrated. Usually teams process more information, knowledge and have more reasoning capacity than an individual. For these reasons, teams are very effective in dealing with large complexity under turbulent scenarios. On the other hand, for timely, simple decision making tasks, individuals may outperform teams (Kang et al., 1998).

(Huberman & Glance, 1998) studied the tendency to defect from a group as consequence of social dilemmas. A social dilemma occurs when an individual in a group can either collaborate on the achievement of the group goal, or defect. The research suggests that a lognormal distribution is a good model to assess the number of defections from a group. However, from the data presented in (Huberman & Glance, 1998 pp 99) a Weibull seems to have a better fit. Besides the type of distribution, another important issue is the clear asymmetry of the model. The results suggest that in order to secure cooperation within a group or organization, members should have small variations in their effort. For the same size, a group with a small variance in their outputs will be more successful at generating cooperation than a group made of individuals with large variation in effort.

Lin studied the effects of different organizational structures with a computational model (Lin, 1998). The research suggests that an organization's ability to achieve a high level of decision accuracy is often achieved at the risk of committing more errors, particularly underestimates. The occurrence of the errors is directly associated with the task environment, the organization's structure and the time pressure. Under scenarios with little or no pressure, organizational design has little impact on performance. Under moderate time pressure the relative advantages between organizational designs become apparent. In such scenarios, simple organizational designs perform the best. Under extreme time pressure there is no advantage to any organizational design (chaos). The research supports one of the fundamental propositions in contingency theory: to achieve

high performance, an organization should adopt an organizational design matched to the task environment.

Woodward has studied the relationship between technological complexity and structure, classifying the technology into three types: (Woodward, 1965)

- *unit,* custom made and non-routine jobs
- *mass*, large batch or mass production in assembly lines
- *process,* highly controlled, standardized and continuous processing, such as refineries

Technology in organizational theory refers to the complete transformation from input to outputs. This concept is more abstract than the idea of physical technology.

This scheme was created for the manufacturing industry and it is not directly suitable for software engineering. However, the characteristics of unit and process technologies: high proportion of skilled workers, low formalization and low centralization, are of interest.

Perrow (Burton & Obel, 1998) introduced a two-dimensional classification of the technology (Fig. 2.5). The first dimension is the analyzability of the problem varying from well-defined to ill-defined. The second dimension is the task variability, which means the number of expected exceptions in the tasks. The scheme lacks a third dimension representing time. Hence, in this projection, software engineering occupies part of the non-routine and part of the engineering regions. During the earlier phases of the development the problem is usually ill-defined. That is why the requirements phase is so

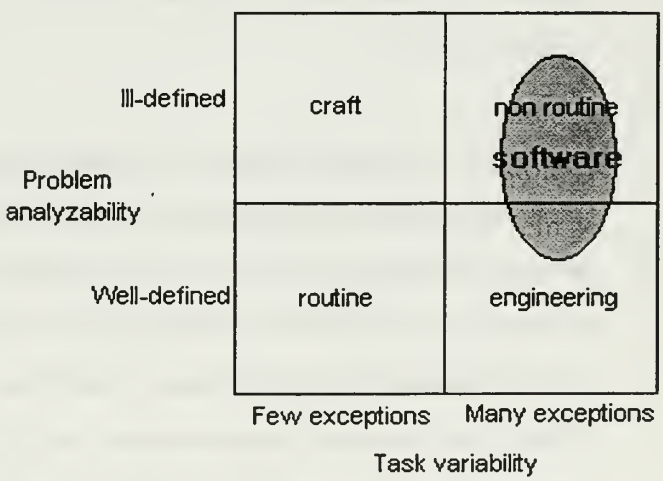


Figure 2.5: Perrow's Classification

prone to errors. After several prototypes and evolution cycles, the problem is transformed into well-defined and the system can be specified. This is a significant difference from other forms of engineering already discussed in Chapter I.

A second line of research (Burton & Obel, 1998 pp. 174-180), introduced a classification based on a four-variable model: *equivocality*, *uncertainty*, *environmental complexity*, and *hostility*. *Equivocality* is "the existence of multiple and conflicting interpretations." It is a measure of the lack of consensus about the framing of the relevant variables and their interrelationships in the space. *Uncertainty* is the lack of knowledge about the likelihood of specific values for the known variables. *Environmental Complexity* is the number of factors and their interrelationships in the environment. Finally, *hostility* is "the level of competition and how malevolent the environment is." In Table 2.8, the fourth variable, hostility, was disregarded because when hostility exceeds a certain threshold, it overrules other factors (Burton & Obel, 1998 pp. 177).

Table 2.8: Burton & Obel's Scheme (Adapted from Burton & Obel, 1998 pp 181-182)

Equivocality	Enviromental Complexity	Uncertainty	Formalization	Organizational Complexity	Centralization
Low	Low	Low	High	Medium	High
Low	Low	High	Medium	High	Medium
Low	High	Low	High	Medium	Medium
Low	High	High	Medium	High	Low
High	Low	Low	Medium	Medium	High
High	Low	High	Low	Low	High
High	High	Low	Medium	Medium	Low
High	High	High	Low	Low	Low

Software development scenarios usually correspond to high equivocality that decreases over time, high environmental complexity and high uncertainty scenarios (dark gray in Table 2.7). These correspond to low formalization and low organizational complexity with centralization inverse to the environmental complexity. The recommended organization could be ad hoc or matrix with coordination by integrator or group meeting. The information exchange is rich and abundant. The incentive policy

group meeting. The information exchange is rich and abundant. The incentive policy should be based on results. These parameters constitute the key points to customize the behavior matrix of VitéProject to software developments.[16]

In another perspective, organizational decision-making can be classified unto four stereotyped models: a) Management Science Model, b) Carnegie Model, c) Incremental Decision Process Model, and d) Garbage Can Model. The characteristics of each class can be summarized as follows (Daft, 1989):

a) *Management Science Model.* This model is analog to the rational approach for individuals. It is based on operational research and has been used with great success to solve military problems as well as business problems. This model work fine when all the variables in the problem, including the environment, are available and measurable, and some rational optimization criteria can be applied.

b) *Carnegie Model.* Cybert, March, and Simon at Carnegie-Mellon University introduced this model based on negotiations. The decisions are made on the basis of coalitions of stakeholders. A coalition is an alliance among several decision-makers about organizational goals and priorities. The coalitions are required to enlarge the political power, to converge the different views into a smaller set of goals and priorities, and to address the cognitive limitations of the individuals. When the problem is ill defined and conflictive, the decision-makers try to find a quick and maybe temporal solution to the problem instead of applying effort to find the perfect and definitive solution. This model is ideal for negotiated decision-making.

c) *Incremental Decision Process Model.* This model introduced by Mintzberg emphasizes the political and social aspects of decision-making. The model assumes that series of small decisions over a period of time produce a major decision. At each small decision the direction is revised, consequently the ultimate decision may be very different from what was initially anticipated.

---

[16] Chapter V, Section C.6 presents the parameter values for VitéProject.

d) *Garbage Can Model.* This model is useful when in highly uncertain scenarios, also called organized anarchies. Organized anarchies do not rely on hierarchies of authority to resolve problems. The approach is more collegial and the solutions tend to be opportunistic and at short term. This model can be applied in what Roberts called *wicked problems* (Roberts, 2000). Such problems are ill defined in terms of the problem definition and solution, they present a poorly understood cause effect relationship among the variables, and the stakeholders exhibit a certain turnover.

Daft proposes a method for choosing the organizational decision-making model base on two indicators: goal consensus and technical knowledge (Daft, 1989). Goal consensus refers to the degree of agreement among decision-makers about the organizational goals and their priorities. In other words, it refers to the problem identification. Technical knowledge refers to the understanding about how to reach the organizational goals, that is the problem solution.

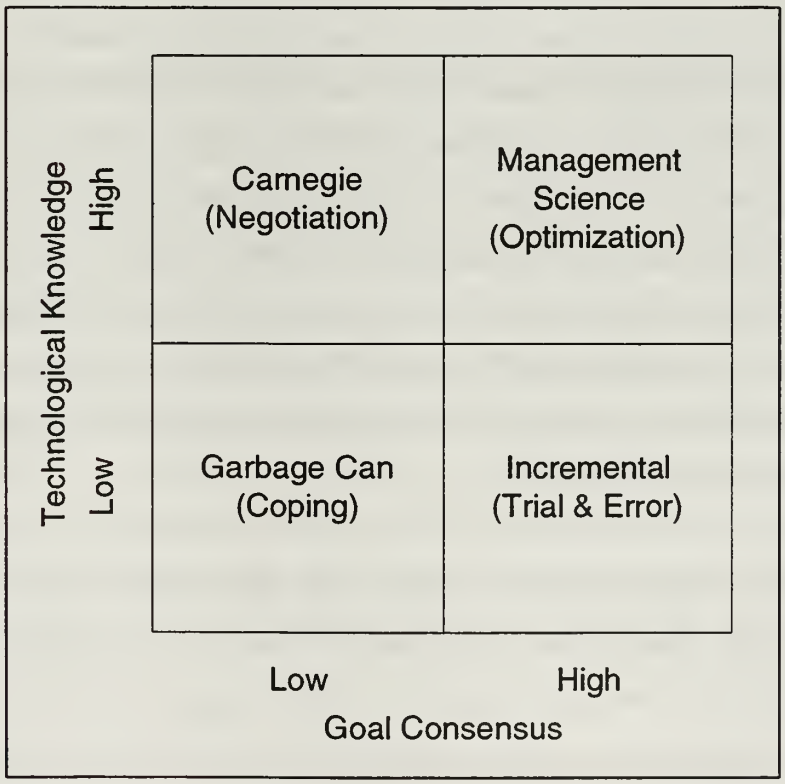


Figure 2.6: Decision-Making Models

During the requirement elucidation, decision-making in software development follows the Garbage Can Model. When the technical knowledge level has been increased but the requirements are conflictive, the decision-making project should adopt the Carnegie Model. If the software project requires research then the Incremental Model could be the best approach. Finally, when the specifications are completed the decision-making model to adopt should be Management Science. Project managers can recognize the situation observing the uncertainty, equivocality, environmental complexity, conflict, and turnover of the stakeholders.

2. The Edge of Chaos

Chaos theory describes a specific range of irregular behaviors in systems that move or change (James, 1996). Chaotic does not mean random. The primary feature distinguishing chaotic from random behavior is the existence of one or more attractors. Without the existence of such attractors the quasi-chaotic scenarios could not be repeatable. It is important to realize that a chaotic system must be bounded, nonlinear, non-periodic and sensitive to small disturbances and mixing. A system that has all these properties can be driven into chaos. The edge of chaos is defined as "a natural state between order and chaos, a grand compromise between structure and surprise" (James, 1996). This concept is closely related to organized anarchies and the Garbage Can Model discusses in the previous section. The edge of chaos can be visualized as an unstable partially structured state of the universe. It is unstable because it is constantly attracted to the chaos or to the absolute order.

People usually believe that "order" is the ideal state of nature. This could be highly inaccurate. Research on organizational theory (Stacey, Nonaka, Zimmerman); Management (Stacey, Levy); and economics (Arthur) support the theory that an operation away from equilibrium generates creativity, self-organization and increasing returns (Roos, 1996). Excessive structural rigidity complicates the adaptation to new

environments. Too much chaos, on the other hand, can make coordination and coherence impossible.

Lack of structure does not always mean disorder. Let's illustrate this idea with an example. A flock of migratory ducks in a lake has little structure. However, a few minutes after they start flying some order appears and the flock creates a V-shape formation. This self-organized behavior occurs because a loose form of structure exists. Experiments with intelligent agents governed by three rules showed the same behavior. These rules are: a) try to maintain a minimum distance from the other objects in the environment, including other agents; b) try to match the speed of other agents in the vicinity; and c) try to move toward the perceived center of mass of the agents in the vicinity), showed the same macro behavior (MIT, 1999). Independently of the starting position of the agents, they always end up in a flock. Even if an obstacle disturbs the formation, the pseudo-order is recovered some time later. This self-organized behavior emerges despite the absence of leadership and without an explicit order to form a flock.

A more interesting example is the behavior of software development teams. A recent article (Cusumano, 1997), describes the strategies of Microsoft to manage large teams as small teams. Cusumano says "What Microsoft tries to do is allow many small teams and individuals enough freedom to work in parallel yet still function as one large team so they can build large-scale products relatively quickly and cheaply. The teams adhere to a few rigid rules that enforce a high degree of coordination and communication." This seems to be a description of the emerging behavior in a complex adaptive system. It is self-adaptive because the agents realize the adjustment to the environment, and it is emergent because it arises from the system and can only be partly predicted. As in the example of the ducks, a few rules of interaction between the agents (in this case *software developers*) generate efficient behavior. The three rigid rules at Microsoft are:

- Daily integration of the work forcing the synchronization and testing of the build

- The developers introducing bugs must fix them immediately and are responsible for the next day's integration
- Milestone stabilizations are sacred

Another possible explanation of Cusumano's observations could be the presence of an underlying structure that propitiates the creativity and productivity.

Complex adaptive systems, as the one just described, are composed of multiple interacting agents. The emergence of the complex behavior requires some conditions. The first condition is the existence of more than one agent. A second condition is that agents must be sufficiently different from each other that their behavior is not exactly the same in all cases. When agents behave in exactly the same way they exhibit predictable, not complex, behavior. Finally, a third condition is required. Complex adaptive behavior only occurs in the edge of chaos.

3. Some of the Risks of Being on the Edge of Chaos

Limiting the structure in organizations can be useful in situations when innovation is critical or when revitalizing bureaucracies is required. However, if the structure is debilitated beyond a certain minimum, it leads to an undesired state. Some traits can alert the eminence of an anarchic situation known as the "chaos trap" (Brown & Eisenhardt,[17] 1998): emerging of a rule-breaking culture, missing deadlines and unclear responsibilities and goals, and random communication flows.

On the other hand focusing on hierarchy and disciplined processes, on schedules, planning and job descriptions may lead to a steady inert bureaucracy. Organizations in such a state react too late failing to capture shifting strategic opportunities. This is the case of a "bureaucratic trap," where there are also some observable warning traits: rule-

[17] Kathleen Eisenhardt is a NPS alumni.

following culture, rigid structures, tight processes and job definitions, and formal communication as the only channel.

The alternative is "surfing" the edge of chaos avoiding both extremes. Achieving this requires limited structure combined with intense interaction between the agents, giving enough flexibility to develop surprising and adaptive behavior. Organizations in this state are characterized by having an adaptive culture. People expect and anticipate changes. A second characteristic is that the few key existing structures are never violated. Finally, real time communication is required throughout the entire organization.

Being on the edge of chaos implies an unstable position where some perturbations can cause the rupture of this delicate equilibrium and the fall into one of the two steady states. A potential perturbation factor is the organizational collaboration style. Too much collaboration can disturb the performance of each agent and consequently, the whole system is affected. On the other hand, too little collaboration destroys the advantage of acting organized and leads to paralysis. Other sources of perturbation are the tendency to be tied to the past and cultural idiosyncrasy, or by contrary, to lose the link with the past. In one case, the change becomes impossible. In the other case, the benefits from previous experiences are not capitalized on. The equilibrium point is called regeneration. In such an unstable state, mutation can occur. Therefore the inherited characteristics that give a competitive advantage in a certain scenario can be perpetuated, and new variations are can be introduced. If too little variation exists, natural selection fails. This regeneration permits complex adaptive systems to change over time following a Darwinian pattern.

Natural selection is an effective, but not generally efficient way to evolve because many errors are committed during this blind process (Brown & Eisenhardt, 1998). The process requires some amount of mutation to avoid the sudden convergence on suboptimal characteristics. Some of the characteristics lost in the past can be reintroduced and can become useful in the new scenario.

4. The Strategic Planning Issue

Traditional approaches to strategic planning emphasize picking a unique strategy according to the competitive advantages of each organization. Porter's five-force approach (Porter, 1980) assumes that there exists some degree of accuracy in the prediction of which industries and which strategic positions are viable and for how long. In a high-velocity scenario, the assumption of a stable environment is too restrictive. Customers, providers, competitors, and potential competitors, as well as substitute products are evolving faster than expected. The introduction of new information physical technology tools, the Internet and the globalization of the markets are contributing to this phenomenon, and nothing seems to reverse the process. The failure of long-term strategic planning is not a failure of management, it is the normal outcome in a complex and unpredictable environment. A growing number of consultants and academics (Santosus, 1998), (Brown & Eisenhardt, 1998) are looking at complexity theory, to help decision-makers improve the way they lead organizations.

How useful could a map of a territory whose topography is constantly changing be? In fast changing environments, survival requires a refined ability to sense the external variables. Traditional approaches rely on strategic planning and vision. However, for unstable planning environments, these approaches would not be effective because it is impossible to predict the scenario's evolution in terms of markets, technologies, customer's needs, etc. Organizations relying only on one vision supported by a tight planning, may not pay enough attention to the future. Consequently, nearly blind, they have little foresight. A certain amount of inertia and commitment to the plans is required to prevent erratic changes caused by reaction diverse variables.

If the available time is shrinking, a different approach is required. The present technological situation can be described as a fast succession of short-term niches. The ability to change and adapt to those niches is the key to success for surviving in such a variable environment. In a systemic approach, the General Systems Theory reveals that

organizations are systems whose viability depends on basic behaviors (von Bertalanfy, 1976). The key elements of this theory are as follows:

- Ability to sense changes in the environment. This is the most primitive form of intelligence. If it is not present, the probabilities of survival are minimal.
- Ability to adapt to a new environment, modifying the internal structure and behavior. The system tries to regulate itself to survive the crisis in hostile scenarios, or to take advantage of the opportunities in favorable ones.
- Ability to learn from the past, anticipating the auto-regulation behaviors and structure before the environment changes. This ability requires intelligence to infer conclusions from the past and apply them to the present.
- Ability to introduce changes in the environment, making it more favorable to the system's needs. In this case, the system has developed the technology (know how and tools) to exert power over the environment. Here the word technology is used in the organizational theory view of the word.

Any mechanical or computing system has some or all of these abilities. These same abilities could be found in any form of life. The more developed the system is, the more of the above characteristics it has. Darwin's Evolution Theory validates this line of reasoning. Natural selection, acting on inherited genetic variation through successive generations over time is the form of evolution. Variation is the way biological systems probe the environment by presenting many alternatives, some of them ending in failure but a few very successful. This process is an inefficient but very effective and robust way of improvement.

Experiments can provide a certain amount of knowledge about the future. In some sense, probes are mutations on a small scale that can cause only small losses. The experiments' results give insights to discover new options to compete in the future and stimulate creative thinking. The research investment pays dividends when a new way of competing is discovered, altering the status quo's rules. When the changes in the environment occur too fast, the variables become more difficult to sense. It is possible

that a sensor was not able to react in time to record the change and transmit the alert. In this case, the system starts to lose information threatening its own viability. When the changes in the environment are too drastic, even if the sensor organs detect the change, the inference organs may not be able to determine an effective course of action because they do not have previous experience to base a decision upon, or because the decision-making process requires more time. This situation also threats the viability of the system in the long run. The effects of drastic variations and high rate of change over systems can be visualized with simple experiments: a) increasing the speed of transmission in a communication channel beyond some limit will provoke the loss of information, b) modifying the pH in the soil beyond a certain limit can cause the death of a plant. The same syndrome can be recognized in any type of organization. It is possible to employ a new strategy. "Competing on the Edge" is a theory which defines strategy as the creation of a relentless flow of competitive advantages that, taken together, form a semi-coherent strategic direction (Brown & Eisenhardt, 1998). The key driver for superior performance is the ability to change, constantly reinventing the organization over time. This factor of success can be applied to software engineering as well as to other decision problems with similar characteristics (see Section.G.5).

If the environment is moving, like in surfing, the best way to remain in equilibrium is by being in rhythm. Successful corporations, such as Intel or Microsoft are in perpetual movement, launching new products with a certain rhythm. Intel is faithful to its founder's (Moore) law: the power of the microprocessors double every eighteen months. Microsoft has a proportional pace on the software sector. The challenges imposed by hyper-competition create similar characteristics in software engineering developments. So, if the rules of engagement prove effective for one discipline, they could prove useful in the other.

5. Application in Software Engineering

Chaos in software development comes from various sources: the intrinsic variable nature of requirements, the changes introduced by new technologies, the dynamics of the software process, and the complex nature of human interaction. These conditions are sufficient for the development of complex adaptive systems where the agents are software developers or parallel collaborative projects. Software development scenarios usually have high equivocality, high environmental complexity and high uncertainty. The suggested organizational structure to deal with such scenarios (Burton & Obel, 1998) should have low formalization and organizational complexity, centralization inverse to the environmental complexity, and a rich and abundant information exchange. The recommended organization should be ad hoc or matrix, with coordination by integrator or group meeting. This organizational style is difficult to achieve when the organizations are large.

A clear solution can be recognized at Microsoft (Cusumano, 1997): parallel developments by small teams with continuous synchronization and periodical stabilization, software evolution processes where the product acquires new features in increments as the project proceeds rather than at the end of a project, and testing conducted in parallel as part of the evolution process. Cusumano observed that small development teams were more productive because: fewer people on a team have better communication and consistency of ideas than large teams, and in research, engineering and intellectual work, individual productivity varies greatly. Software development requires teamwork. So it is necessary to understand the dynamics of organizations as artificial social entities that exist to achieve a specific purpose, in this case to develop software. Such organizations are composed of individuals who accomplish diverse desegregate activities that require coordination and consequently information exchange.

In order to apply Microsoft's approach three factors should be resolved. First, automated risk assessment is required (the topic of this research). Second, each evolutionary software process should have a maximum evolution speed. If the evolutions occur too fast, without a period of relaxation, it is certain that the process will fall into

chaos (see Chapter VII, Section B). On the other hand if the speed is too slow, then the productivity could decline. The correct rhythm for software processes has not been researched and remains in the hands of the project manager. Third, software processes should be focused on *flexibility* and *extensibility* rather than on *zero defects*. This assertion sounds alarming. However, it is necessary to prioritize the speed of the development over zero defects. Extending the development in order to reach the highest quality could result in a late delivery of the product, when the opportunity niche has disappeared. This paradigm shift is imposed by the hyper-competition.

A shift from the traditional long-term development organizations is required. Virtual teams created as temporary dynamic project-oriented structures, with a composition of skills matching the objectives exactly could improve the current performances. Such virtual organizations are not exposed to bureaucratic loads and need not absorb the cost of permanent staff (Senegupta & Jones, 1999). Larger developments could be achieved by parallel projects loosely coupled sharing a common architecture, such as CORBA or DCOM. This paradigm enables the possibility of managing large developing organizations as if they were small. In such scenarios, the benefits of complex adaptive systems will occur at two levels: at the micro level, inside each small project, the agents are individuals; and second, at the macro level where the agents are the small parallel projects.

Recently Beck introduced a methodology based on chaos called Extreme Programming (XP). Extreme Programming (XP) is a lightweight methodology for small-to-medium-sized software development teams (two to ten programmers), that is specially suited for projects with vague or rapidly changing requirements (Beck, 1999). The method promises to reduce the project risk, improve responsiveness to business changes, and improve productivity throughout the life cycle. XP requires tools to automate testing and integrate buildups in a fraction of a workday.

XP proposes that the classic rule "the cost to fix a problem in a piece of software rises exponentially over time" is obsolete. There are two reasons for that. The first reason

is related to the chaos theory. In a rapidly changing environment the traditional way of planning may result inadequate. Therefore, it could be more economic to deliver the product sooner even if some corrections must be done later. The traditional alternative of delivering the full functionality requires more time, hence when the product is delivered the environment could have changed, making the product obsolete. This controversial position is supported by (Beck, 1999).

XP practitioners present a second reason to contradict the rule, which comes from the technical field. Object-oriented programming (OOP) and object-oriented design (OOD) enable a quicker and cheaper modification of code. Instead of being careful to make big decisions early and little decisions later, it is possible to approach the problem making decisions quickly relying on automated tools to improve the design. This argument has two weaknesses:

- It is true that OOP-OOD facilitate maintenance. However, choosing the wrong architecture could lead to a disaster. So, it is not true that all the decisions can be quickly done.
- The approach requires the use of an object-oriented database (OODB). Without OODB any modification in the model classes imposes changes at the layer of classes dealing with persistence.

XP practitioners claim that software development processes fail to deliver value because of lack of flexibility. The following list presents the problems that characterize the present state of the art techniques, and the way XP addresses them.

- Schedule slips. XP proposes to reduce the length of the release to a few months at most. Each release is focused only on a subset of high priority requirements. Each release is composed of a succession of evolutionary cycles from one-to-four-week length.
- Project cancellation. XP tries to avoid project cancellation by narrowing each release to the smallest set of requirements that makes the most business sense.
- Costly maintenance. XP does not provide explicit solutions to this problem. It relies on the use of project databases and tools.

- Defect rate. XP proposes to reduce the defect rate by intensive testing and inspections. Testing is conducted by developers and customers. Developers write test scripts for each function. Customers write test scripts for each feature. XP requires a testing tool capable of executing the scripts. XP uses intensive code inspections by imposing the unusual practice of working in pairs at each workstation.
- Business misunderstandings. To avoid the communication problem between customers and engineers, XP proposes the incorporation of a group of customers in the development team. This requirement imposes a severe constraint because the customer usually cannot provide full time delegates.
- Lack of quick responses to business changes. The reduction of the scope for each release provides the responsiveness required. This is a key aspect of the methodology, instead of using a traditional long term planning, XP uses a quasi chaotic process in which a succession of small releases are delivered reacting to the changes of the environment.
- Staff turnover. Despite the intentions, XP cannot provide a real solution for this problem. The recommended actions about estimations made by developers, human communication, and work environment are too weak. As in any other methodology, staff turnover is a problem that can be mitigated but not resolved.

Despite being a methodology with limitations in terms of scalability, XP introduced important contributions:

- The recognition of uncertainty as an inherent characteristic of software projects
- The identification that a software process based on quasi-chaotic behavior is a good method to deal with uncertainty. Lots of small inexpensive corrections and feedback can produce a better result than the traditional approach.

6. Conclusion

Complex adaptive systems appear to be the most attractive way to deal with changing environments. Besides some indicators introduced by (Brown & Eisenhardt, 1998), the academic research is not advanced enough to assert a methodology for competition on the edge. Some enterprises like Microsoft and Intel have discovered and applied this form of strategy for many years, but little information has permeated. The drastic change proposed in the software processes aims to use the benefits of programming in the small to programming in the large. Moreover, the quality-driven paradigm should be revised, and the objective should be shorter delivery times, flexibility and expansibility.

THIS PAGE INTENTIONALLY LEFT BLANK

III. CONCEPTUAL FRAMEWORK

This chapter contains the framework to risk identification and risk assessment. Causal analysis was used to find the primitive origins of threats in a project, and to find a way to identify and to assess risk automatically. From the point of view of software engineering, it was necessary to create the methodology to frame the decision-making process during the early stages of the life cycle, when changes can be made with less impact on the budget and schedule. Field found that the most significant causes of project failures are: lack of understanding of user's needs, ill defined scopes, poor management of project changes, changes in the chosen technology, changes in the business needs, unrealistic deadlines, user's resistance, loss of sponsorship, lack of personnel skills, and poor management (Field, 1997).

Risk management can be divided into three activities: risk identification, risk assessment and risk resolution. Risk identification is the set of techniques designed to alert and identity possible threats. Risk assessment is the quantitative analysis of the probabilities and impacts of the identified threats. Risk resolution is the application of both resources and effort to avoid, transfer, prevent, mitigate or assume the risks. This third activity, risk resolution, is beyond the scope of this research.

In order to achieve risk management, an organization requires a minimum level of maturity that can be associated with CMM level 2. SEI followers said that "many organizations are unable to manage risks effectively for any of the three following reasons: a risk-averse culture; an inadequate management infrastructure to support effective risk management; or the lack of a systematic and repeatable method to identify, analyze, and plan risk mitigation" (Carr, 1997). If an organization is not able to collect

metrics, any attempt to formally identify and assess risks is impossible. Project managers require critical information to make timely and prudent decisions. It is not surprising that increased complexity can decrease a project manager's ability to identify and manage risk.

In this research vision, software risks could be controlled if the problems of how to administrate uncertainty, complexity and resources were solved. Transforming the unstructured problem of risk assessment leads to a formal method able to be translated into an algorithm. In order to structure the problem, project risk was analyzed and decomposed into simpler parts. Using causal analysis three major risk contributors were identified: *resource risk*, *process risk*, and *product risk*. Each of these factors introduces risks individually and due to their interactions.

Resource risk is affected by organizational, operational, managerial and contractual parameters, such as outsourcing, personnel, time and budget among other resources. The literature is abundant in this area (Hall, 1997), (Karolak, 1996), (Grey, 1995). Various approaches use subjective techniques, like guidelines and checklists (SEI, 1996), (Hall, 1997), (Karolak, 1995), which when even supported by metrics, require experts' opinions.

Engineering development work procedures, such as software development, planning, quality assurance, and configuration management cause *process risk*. The more complex a process is, the more difficult it is to manage, and the more education, standards, reviews, and communication are required. Consequently, complexity grows. The software process complexity has been partially covered by research in terms of subjective assessments about maturity levels and expertise (SEI, 1996), (Hall, 1998), (Humphrey, 1989). However, a more precise and objective method is required. Several approaches to study process complexity have been introduced in the field of systems management. Nissen introduced a method that measures the complexity of processes based on the characteristics of the graph that model the process (Nissen, 1998). Abdel-

Hamid introduced a method based on simulation to study the process dynamics (Abdel-Hamid, 1989 and 1991).

Finally, *product risk* is related to the final characteristics of the product, its complexity, its conformance with specifications and requirements, its reliability and customer satisfaction. The product introduces its own threats terms of quantitative and qualitative attributes. Two basic product-risk factors: *requirement volatility*, and *complexity* were identified. *Requirement volatility* refers to the speed of changes in the requirements. This measure shows the difficulty of the requirement elucidation process.[18] High volatility is characteristic at the beginning, when the problem is ill defined. The requirement volatility can be measured from the requirements baseline.

The concept of *complexity* used in this dissertation emphasizes the relational notion of complexity. In general the complexity of an object is a function of the relationships among the components of the object. In an early vision of a modern object oriented paradigm, (Myers, 1976) introduced three valuable concepts to measure complexity:

- *Independence*: The independence of each component can reduce the complexity of the system if the components are a partition of the system. So, there is maximum cohesion and minimum coupling.
- *Hierarchy*: Hierarchical structures allow the stratification of the system in different layers of abstraction.
- *Explicit communication*: The components should communicate with explicit protocols avoiding any hidden side effects.

The concept of *functional complexity,* commonly used in software, refers to the relational but also to a rational notion related to cognitive difficulty. The computation of the relational complexity during early phases can be done from formal specifications, as discussed in Chapter V.

---

[18] This is the same definition used in VitéProject (Chapter II, Section F.1).

The analysis showed a dependency between these classes of risk. Dependencies between process, resources, and product constitute an equivalence relation (Fig. 3.1) because the symmetric, transitive and reflexive properties apply. Moreover, the three classes are one equivalence class in the relation. The strong dependency between the three concepts reflects the vision that resources, process and product are different facets of the same entity: the project.

The process provides the description of its environment and the theoretical requirements to execute it. The resources represent the actual allowances in personnel, tools, budget and schedule. Consequently, the process and the resources introduce threats due to the mismatches between the process's characteristics (complexity, technology required, budget required, schedule required, and personnel skills required) and the actual resources available.

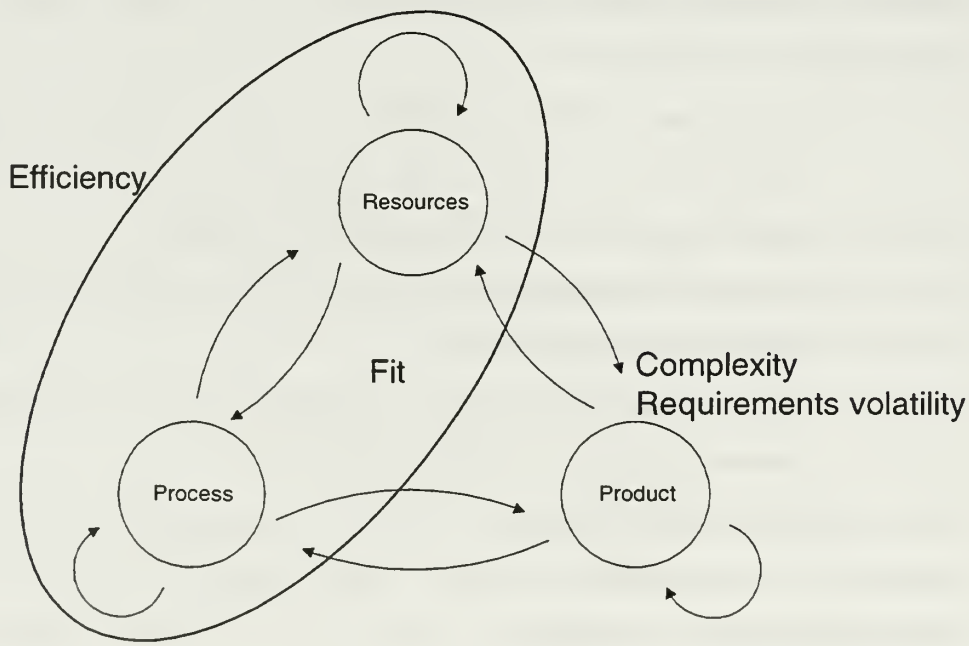


Figure 3.1: The Equivalence Relation

The interaction between the process and the resources defines *the organizational fit*. If there is a perfect match between the process and the resources, it is expected that the *efficiency* of the project will reach its maximum (Fig. 3.1). By contrast, if mismatches between the process and the resources exist, then it is expected that the efficiency will decrease. This observation has been proved in previous research (Burton & Obel, 1998).

The product introduces threats as consequence of its *complexity* and its *requirements volatility* (Fig. 3.1). The correlation between complexity and size has been showed by (Albrecht, 1979 and 1983). More size implies more time, effort, cost and defects (Boehm, 1981). Changes in requirements are inevitable. They express the difficulty in requirements elucidation. A change is not directly related to an increase in the complexity. It can also reduce the complexity. For that reason complexity and requirements volatility are independent metrics that explain different aspects of the product.

The decomposition created by causal analysis (Fig. 3.2) revealed a method to identify risks by comparing the degree of mismatching between the product and process characteristics, against the resource constraints. Causal analysis also revealed candidate indicators to be used in the estimation model. Chapter V will introduce three groups of metrics: for requirements, for efficiency, and for complexity. These three groups of metrics correspond to the three risk factors identified by causal analysis.

Figure 3.2 presents the fish diagram representing the cause-effect relationships in the risk framework. The rectangles represent the concepts discovered and interrelated by cause-effect arrows. The shadowed rectangles represent the areas covered by this dissertation. The uncovered ones represent opportunities for future research. Ovals represent the metrics associated with the concepts. The project manager as the decision-maker must decide between different alternatives each of them with different associated risks. To deal with risk, the decision-maker should consider also various *external*

constraints like the time frame, the cost frame, the expected return of investment, the cost of failure, and his *decision preferences*.

The project risk is caused by two components: the *fit*, which is measured in terms of *efficiency*; and the product, which depends on its requirements and functionality measured in terms of *requirements volatility* and *complexity* respectively. Note that the project risk has two associated metrics: *time* and *probability of success*. The probability of success is a function of time. Conversely, given a certain amount of time, one can estimate the probability of success of the project.

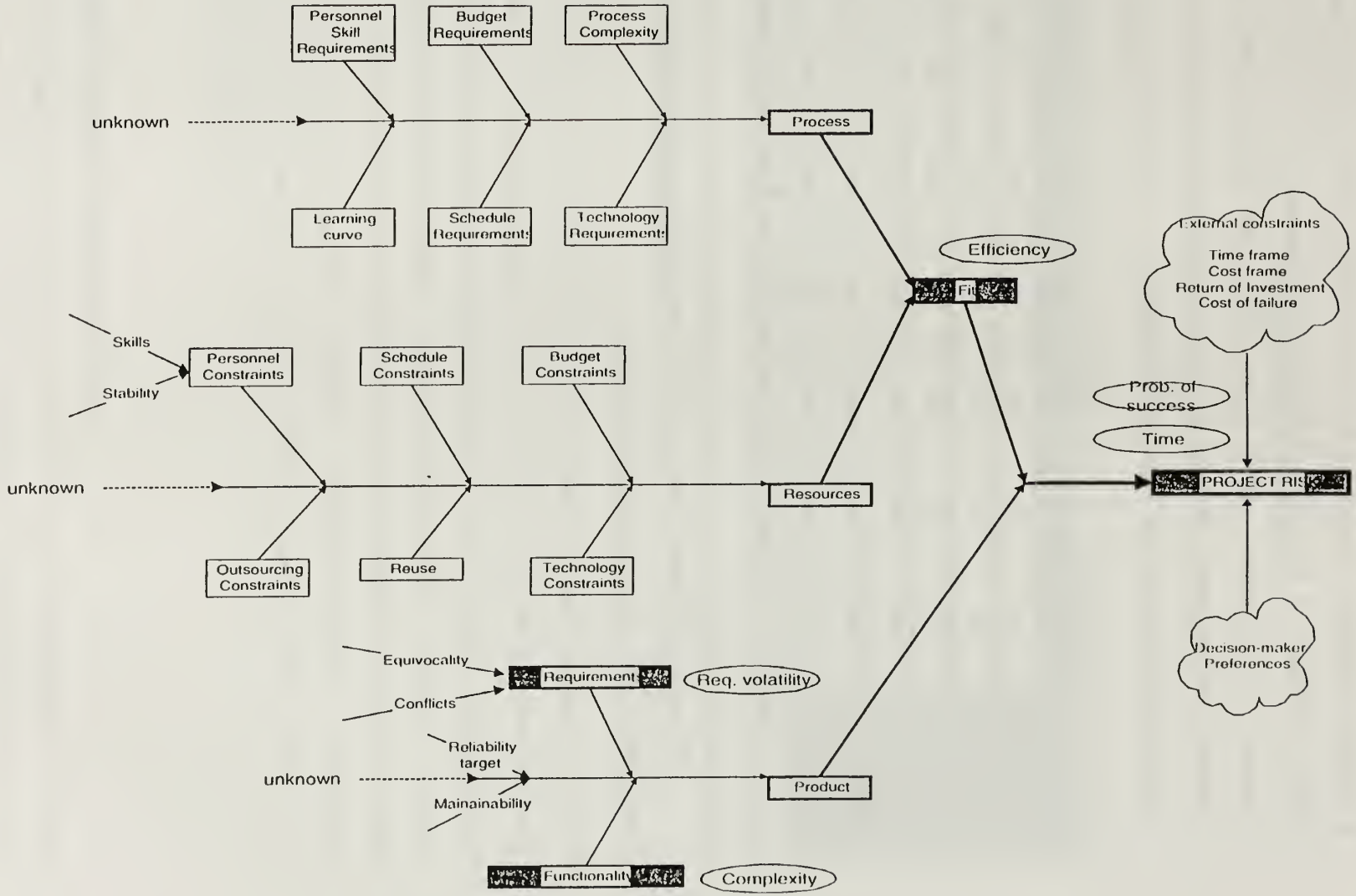


Figure 3.2: Causal Analysis Fishbone Diagram

The framework previously discussed is original and independent of other research. It has been validated by comparing the main concepts, discovered using causal analysis, to the main lines of research available in the literature (SEI, 1996), (Hall, 1997), (Charette, 1997), (Boehm, 1989), (Jones, 1994), and (Karolak, 1996). Figure 3.3 summarizes the comparison between the author's approach and the references. The details of these approaches have been presented in Chapter II.

Concept	Nogueira	SEI	Hall	Charette	Boehm	Jones	Karolak
Req. Volatility	yes	yes	yes	yes	yes	yes	yes
Efficiency	yes	yes	yes	yes	yes	yes	yes
Complexity	yes	yes	yes	yes	yes	**no**	**indirectly**
Time	yes	yes	yes	yes	yes	yes	yes
Probability	yes	**no**	**no**	**no**	**no**	**no**	**subjective**
Formal model	yes	**no**	**no**	**no**	**no**	**no**	**almost**

Figure 3.3: Comparison between the Different Risk Methodologies

The main difference of this approach is the emphasis on using formal models and probabilities rather than subjective assessments.

IV. RESEARCH DESIGN

A. INTRODUCTION

The problem of productivity is partially solved. The industry has tools that improve software development productivity. New efforts in this vein are not the solution for the software crisis because the problem in the author's opinion is focused on organizational and human communication issues. Software development is still a human dependent activity requiring vast human communication, and without appropriate managerial decision support tools, software engineering will remain in its present state. A better understanding of the knowledge about the internal phenomenology of the software life cycle is required to improve software development because it is in the human aspects of the software process where the bottleneck is located now. Without such knowledge, risk assessment is almost impossible.

As outlined in the introduction, this research focuses primarily upon risk assessment for software engineering. More precisely, it addresses the issue of human dependency in risk assessment of the evolutionary software processes incorporating an automated risk assessment method. Despite the improvements achieved in software processes, software reuse and automated tools, risk assessment for software projects remained an unstructured problem dependent on human expertise. This research transforms risk assessment into a structured problem using indicators measured in the early phases of the project.

B. PRIMARY RESEARCH QUESTION

The primary research question is: *What are the early automatically collectable measures from the software process that describe project risk?* The risk of the project is

related to its probability of success. That is the probability of reaching the objective with the assigned resources in the allocated time. The main point in the question is the discovery of a set of good indicators for risk. These indicators should be recognized during the early phases of the process in order to provide early alert. To answer the research question the following activities were required:

1. Review the literature about software evolution. This study helped the author understand the scope and limitations of evolutionary software processes, and it helped to define and narrow the problem.
2. Review the literature about risk management from the operational research point of view. This study provided a theoretical background to produce a mathematical model.
3. Review the literature about risk management in the field of software engineering. This study showed two well-defined groups of researchers. The first group follows a less rigorous and human dependant approach starting from the beginning of the project. The second group, which corresponds to the software reliability field, follows a rigorous approach post mortem. This second approach provided insights in how to link the operational research methods with the software engineering approaches.
4. Employ causal analysis to find a set of candidate indicators for risk. The set of candidate indicators was compared to notorious frameworks for risk to check their consistency (see Fig. 3.3). It was found that requirement volatility, organizational efficiency, and product complexity were promising indicators.
5. Review the software economics research, especially COCOMO and Putnam's models. This study showed that the estimation models available today have some limitations when applied to evolutionary software processes (see Chapter II, Section D).
6. Conduct experiments to prove the correlation between complexity and size by using the available baselines of projects created by the evolutionary software process, specifically using CAPS and PSDL.

C. SECOND RESEARCH QUESTION

The second research question is: *How can these measures be related in order to assess project risk?* Answering this question implies the formalization of a model and its calibration and validation in three ways: a) internal consistency proved by mathematics and statistics; b) black box validation by comparing its outputs in duration and effort with other available models; and c) black box validation against a set of observations. To achieve this last goal a large set of well-measured software projects is required. This set has not been found yet. A second and more promising alternative was to simulate a set of projects. VitéProject was chosen as the simulation tool for the following reasons:

- Availability
- Possibility of customizing and controlling parameters
- Inclusion of communications and exceptions in the model
- Given that the proposed model uses parameters collected during the early phases and given that VitéProject requires a complete breakdown structure of the project, which can be done only in the late phases, a considerable time gap between the two measurements exists. Such a time gap is less than conducting a post-mortem analysis, but is sufficient for calibration and validation purposes.

Due to its generic design, VitéProject had to be configured for the specific domain. To solve this problem, it was necessary to review organizational theory and use an expert system (Organizational Consultant) to obtain the correct parameters (see Appendix A).

D. VALIDATION

According to Cook, Stanley and Campbell, the experiments conducted in this research are *true experiments* (Cook & Campbell, 1976). The use of true experiments facilitated the validation process because the random sampling minimized the occurrence of the confounding factors so common in *quasi-experiments*. The typical obstacles in the use of true experiments (Cook & Campbell, 1976) were resolved as follows:

- *Lack of randomness in the sampling process.* This issue was controlled by the use of random algorithms in VitéProject.
- *Faulty randomization procedures.* This issue was addressed relying on the validation of VitéProject (see Chapter II, Section E).
- *Sampling variability.* This issue was addressed by using large samples according to the Central Limit Theorem (Devore, 1995).
- *Treatment-related refusals from individuals to participate in the experiment.* This problem is common in social sciences, but does not affect this research.
- *Treatment-related attrition from experiment.* As the previous point, this problem was avoided by the use of simulations.
- *Heterogeneity in the extent of treatment implementation.* The use of algorithms guaranteed the same treatment for all the experiments.
- *Confounding factors in the control group.* Simulations controlled this problem.
- *Treatment contamination.* Simulations controlled this problem.
- *Confounding factors due to responses from individuals.* This problem did not exist because of the use of simulations.

Cook and Campbell introduced an enhancement to the previous work of Campbell and Stanley (Cook & Campbell, 1976). This new scheme is based on four types of validity:

- *Internal Validity.* This refers to conclusions that one can draw about whether a demonstrated statistical relationship implies cause. It is a deductive process in which the investigator needs to think whether or not all the confounding factors can be ruled out. One of the threats to the internal validity is the lack of control over the samples and the selection of the samples. The internal randomization and controllable parameters in the simulation are the key factors that contributed to validate internally the models. In the case of this research the internal validity is maximized by the use of true experiments (Cook & Campbell, pp. 230).

- *Statistical Validity.* This refers to conclusions that one can draw based on statistical evidence. This research used extensive statistical evidence to assess the validity of experiments. Chapters V, VI, and Appendix D, present the statistical evidence.
- *External Validity.* This refers to the validity with which a causal relationship can be generalized across persons, settings, and times. This process requires random sampling, heterogeneous groups, or modal samples (Cook & Campbell, 1976). In the case of this research the techniques employed were *random sampling, heterogeneous groups,* and *modal sampling.*[19] Random sampling is the most powerful method for external validation (Cook & Campbell, 1976 p 237). Random sampling enabled to obtain representative samples. This could be done because the internal randomization of the simulator. The samples observed in our experiments were randomly selected by the internal algorithms of VitéProject. We used heterogeneous groups with high and low efficiency.
- *Construct Validity.* This refers to the possibility that a cause or effect can be constructed in terms of more than one construct. Construct validity guarantees the ability to generalize cause-effects relationships in the model. The way to obtain construct validity is by causal analysis and by using true experiments. The main threat to construct validity comes from the investigator's subjectivity, for that reason double-blinded experiments are recommended (Cook & Campbell, 1976 p 239). The construct validity requires a rigorous definition of potential causes and effects. The causality was studied analyzing the results of experiments with random sampling. This research relied on algorithms and expert systems to assess the value of the different simulation parameters. In this way, the role of the investigator was limited to alter three inputs and to observe one output. We applied sensitivity analysis to study the effects of the diverse components in the models and their interaction.

[19] Modal sampling refers to customizing the samples according to a particular area of study, in this case software development.

The validation process for this research is based on the integration of several experiments and software systems as shown in Fig. 4.1. The dotted areas show different validated components of this research. These components were linked together by overlapping validations.

- *The component number 1* corresponds to the conversion from PSDL code to the complexity measure LGC. This process was conducted using experiments, and was statistically validated (see Chapter V, Section A.3). The use of true experiments and algorithms satisfies the internal, external and construct criteria (see Fig. 5.3).
- *The component number 2* corresponds to the conversion from LCG to Ada LOC. The validation process is similar to above (see Fig. 5.4).
- *The component number 3* corresponds to the conversion from size to time. The validation was conducted by comparing it to the COCOMO and Putnam models (see Chapter V, Section C and Fig 5.6).
- *The component number 4* corresponds to OrgCon. The validation process of OrgCon was external to this research. The description of the validation process is presented in Chapter II, Section E. Orgcon was used to assess three parameters of VitéProject: *centralization*, *formalization*, and *matrix strength*, according to the characteristics of a typical software organization (CMM 2 or 3). The values for these parameters remained constant for all the experiments. The use of an expert system, instead of a subjective estimation, provided a formal way to assess these parameters.
- *The component number 5* corresponds to VitéProject. This validation is also external to this research. The description of the validation process is presented in Chapter II, Section F.

In a macro view, this system has three variable inputs (represented by shaded rectangles), and one output (Vité Simulated Time). The shaded rectangles represent the three risk factors: *complexity*, *efficiency*, and *requirements volatility*. These risk factors

were derived from casual analysis (see Chapter III). The metrics related to these risk factors are presented in Chapter V. Let's describe how these risk factors influence the parameters for VitéProject.[20]

- *Complexity*. The complexity level is directly reflected on the parameter *solution complexity*. This parameter has a discrete range of values (low, medium, and high). Complexity also affects indirectly the parameter *FTE* (effort). This is a real-value parameter. To avoid subjective assessments of the effort, the FTE parameter was calculated using the conversion PSDL-time. This conversion is a three-step process. First PSDL was converted into LGC (component 1 Fig 4.1). Then LGC was converted into LOC (component 2 Fig. 4.1). Finally, LOC were converted into time (component 3 Fig. 4.1).
- *Efficiency*. The efficiency level has a direct impact on the parameters *team experience*, *application experience*, and *skill levels*. The possible values for these parameters are discrete (low, medium, high).
- *Requirements Volatility*. Requirements volatility affects the parameters *uncertainty* and *requirement complexity*. The possible values for these parameters are discrete (low, medium, high). Note that requirements volatility measures the difficulty in elucidating the requirements.

[20] Table 5.1 shows a summary of the values used for VitéProject's parameters.

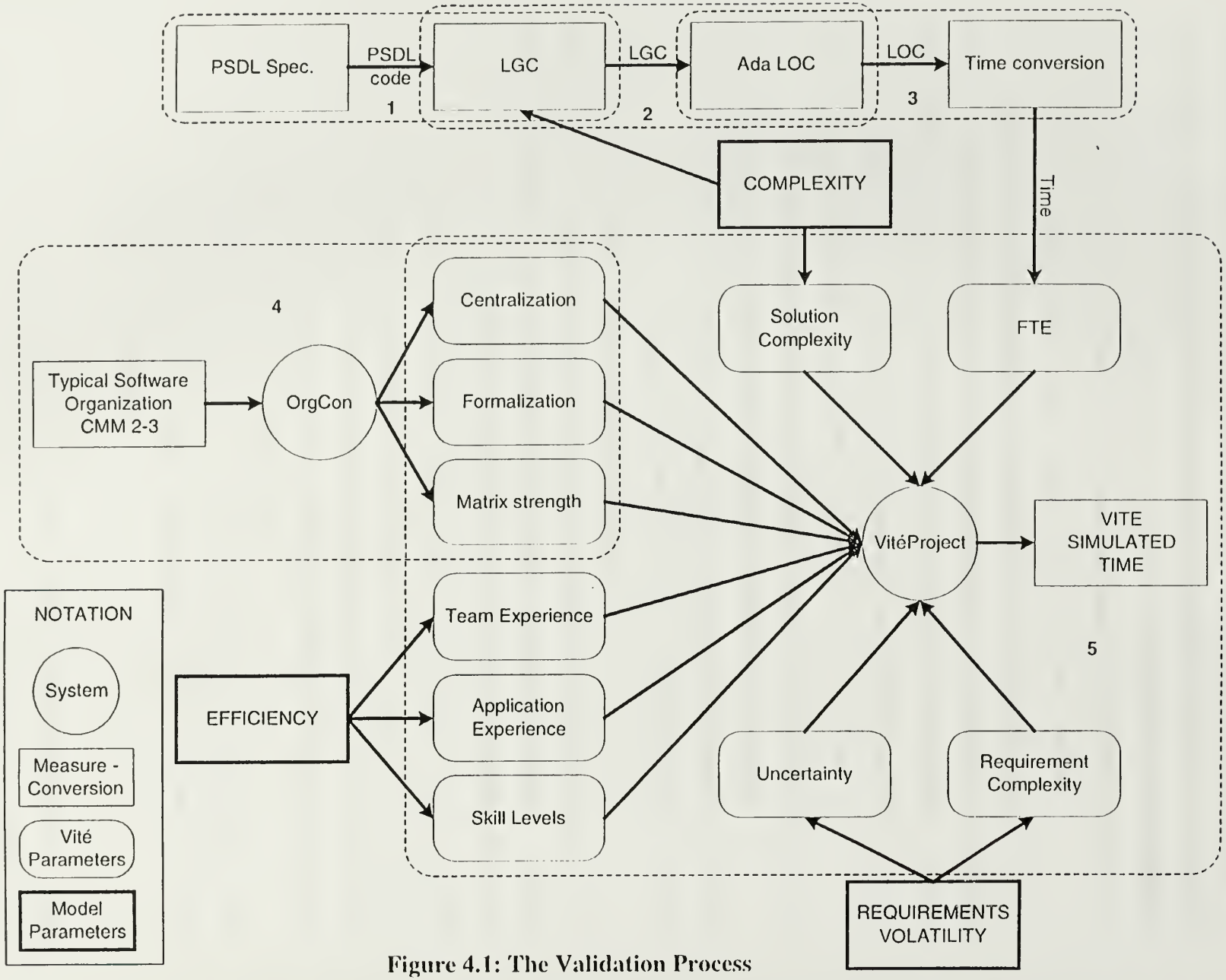


Figure 4.1: The Validation Process

In conclusion, the validation strategy for this research follows the precepts described in (Cook & Campbell, 1976) and (Thomsen et al., 1999).

- *Representational Validity*: The accuracy in representing reality, in this case software development projects, was proved by comparing this framework to other research (SEI, 1996), (Hall, 1997), (Charette, 1997), (Boehm, 1989), (Jones, 1994), and (Karolak, 1996) (see Fig. 3.3).
- *Internal Validity*: The subjectivity introduced by the investigator was controlled by the use of true experiments. The randomness and automatism of the procedures of collecting data impedes the introduction of any subjectivity.
- *Construct Validity*: The use of true experiments, random samples, and controlled simulations assures the construct validity.
- *Statistical Validity*: This is the main validation method applied in this research. Chapter VI and Appendix D contain the statistical analysis.
- *External Validity*: External validity was assured by the use random sampling, heterogeneous groups, and modal samples. External validity was also tested by comparing the results with other available estimation models.
- *Reproducibility*: This form of validation refers to the possibility of reproducing the observed results for different humans. The use of automated methods to collect metrics and software simulations assures that the experiments of this dissertation are reproducible.

E. SUMMARY

Figure 4.2 shows the components of this dissertation as blocks of a pyramid. The arrows show dependencies gluing the blocks. Such dependencies have been validated according to Section D. The contributions are represented in bold fonts. The foundation for this research was extensive literature research of software engineering, software reliability, risk management, organizational theory, chaos theory, and statistics. The heart of the research contains three probabilistic models for risk assessment. These models are supported by three pillars:

a) The framework for risk (Chapter III).

b) The statistical analysis of the results of simulations conducted using VitéProject and OrgCon (Chapter VI and Appendix D).

c) The experiments over real projects developed with CAPS (Chapter V).

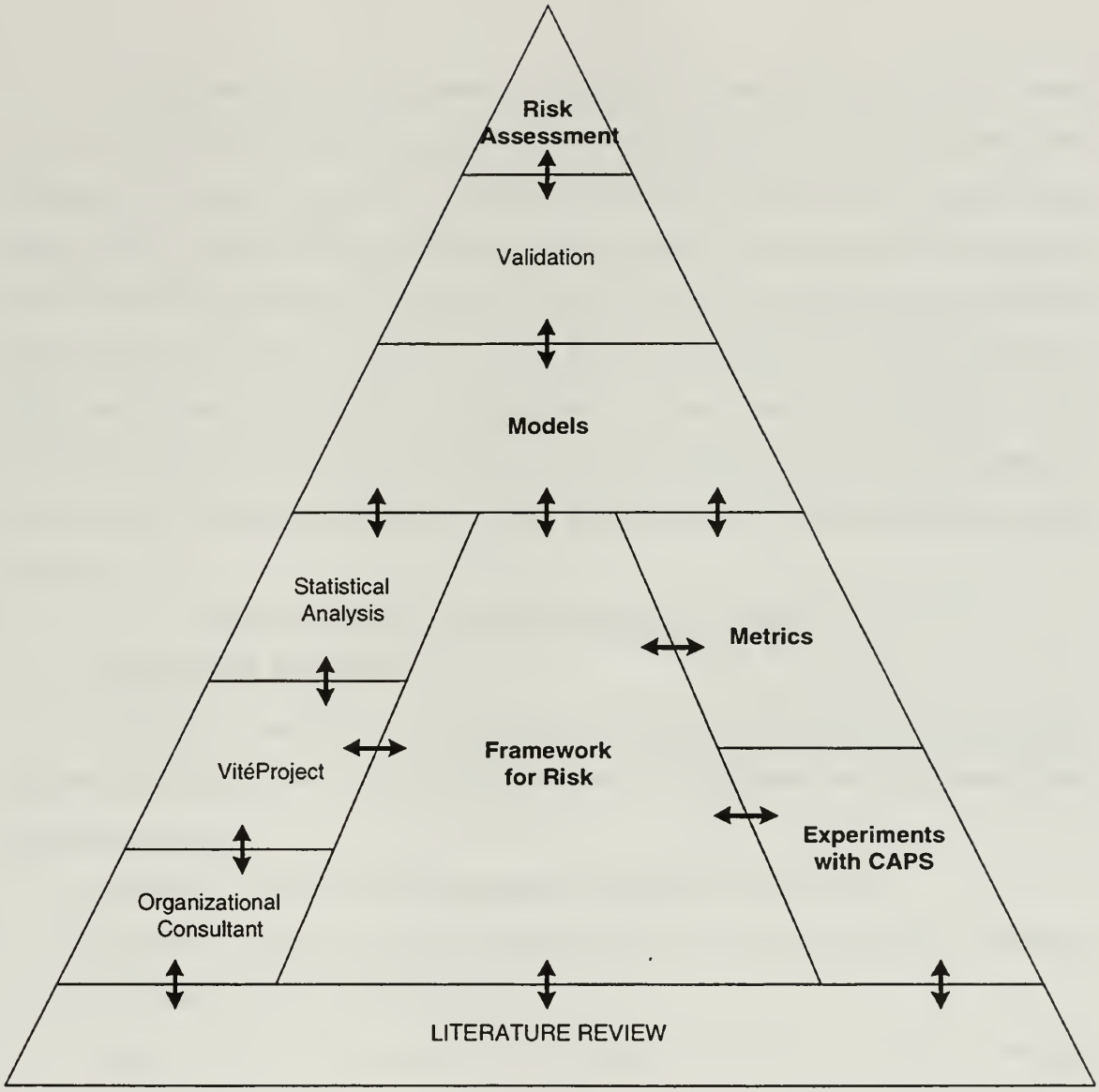


Figure 4.2: Pyramid of the Research

THIS PAGE INTENTIONALLY LEFT BLANK

V. DEVELOPMENT OF THE MODEL

This chapter applies the framework described in Chapter III to develop a model for risk identification and risk assessment. Some concepts about software metrics are discussed, presenting a small set of metrics in which the model will be based. These metrics can be collected from the beginning of the project. This characteristic addresses the issue stated by (Fenton & Pflegger, 1997 pp. 447): "Most general models perform poorly because they were developed from a post hoc analysis of a particular data set. They incorporate the particular characteristics of the data, often including input parameters that would have been difficult to assess accurately at the start of the project. Thus, the first step in building better models is taking care to include parameters known early on."

A. SOFTWARE METRICS

This section describes a set of metrics that support the risk identification strategy. All the metrics presented here are well formed, in the sense that they present the following strengths:

- *Robustness*. Capacity of being tolerant to variability of the inputs.
- *Repeatable*. Different observers would arrive at the same measurement, regardless how many repetitions take place.
- *Simple*. Using the least number of parameters sufficient to obtain an accurate measurement.
- *Easy to calculate*. They do not require complex algorithms or processes.
- *Automatically collected*. There is no need for human intervention.

Metrics are a key factor in the identification of threats. Without metrics, providing early alerts of risks is impossible. There are some erroneous perceptions about metrics that are necessary to clarify:

- *Metrics act against the creative process.* This is an excuse to avoid the use of metrics. Metrics should be collected without the direct intervention of humans. The collection process should be transparent to designers.
- *Metrics represent an additional work load.* The collection procedure can be automated, so the extra workload is not significant. The analysis of the metrics requires the attention of the project manager, and this is his or her normal work.
- *The benefits of metrics are unclear.* This myth is really irrational. Without measures over the process, it is impossible to assess how much effort is required, or what risks should be mitigated.
- *People are afraid of metrics.* This is true, and it is very common to find some resistance to the introduction of a metrics plan. It is important to use the metrics to measure the process rather than use them to punish low productivity.

The minimal set of metrics to support the risk assessment model cover three areas: a) for requirements, b) for efficiency, and c) for complexity. These three groups of metrics correspond to the three risk factors that were identified by causal analysis, described in Section 3.1 (see Fig. 3.2).

1. Metrics for Requirements

a. Birth-Rate (BR)

Birth-rate is defined as the percentage of new requirements incorporated in each cycle of the evolution process. This metric shows the explosion of new requirements as a percentage.

$$BR = (NR / TR) * 100\ (\%) \qquad \text{[Eq. 5-1]}$$

where NR = number of new requirements

TR = total number of requirements = PR + NR

PR = previous requirements

b. Death-Rate (DR)

Death-rate is defined as the percentage of requirements that are dropped by the customer in each cycle of the evolution process.

DR = (DelR / TR) * 100 (%) [Eq. 5-2]

where DelR = number of requirements deleted

TR = total number of requirements (before deletion) = PR + NR

c. Change-Rate (CR)

Change-rate is defined as the percentage of requirements changed from the previous version.

CR = (ModR / TR) * 100 (%) [Eq. 5-3]

where ModR = number of requirements changed

TR = total number of requirements

From the point of view of the metrics, a change in a requirement can be viewed as a death of the old version and a birth of the new one. This simplification does not imply losses of information about the history of the evolution. Requirements volatility expresses how difficult the requirement elucidation process is. The requirements volatility is obtained by the following formula:

RV = BR% + DR% [Eq. 5-4]

For instance if BR = 20% and DR = 10% then RV = 30%.

The traceability of the evolution remains in the hypergraph model. The simplification just described enables one to compare the birth-rate and death-rate in a bi-dimensional plot that shows four regions: *stability region*, *growing region*, *volatility region*, and *shrinking region* (Fig. 5.1). The graph is double logarithmic, so the borders of the four regions are in the 10% value. Each of these regions has different risk connotations because they present different levels of difficulty in terms of the elucidation of requirements. The arrow shows the normal evolution of the project as the time goes by. During the early stages, it is normal for projects to be in the growing region. However,

if the project continues in this region after many cycles, or returns to this region after visiting other regions, a problem has occurred. In the first case, this is an indicator that the requirement engineering is not efficient; hence some corrective action should be applied. The second case shows evidence of late discovery of some cluster of hidden requirements.

After some cycles, the project should be in the volatile region. If the project does not evolve through the stability region, then there is evidence that the requirements engineering activity is not being efficient and some corrective action is mandatory. It is important to analyze the evolution of the stakeholders' issues and criticisms. It may also be the case that the stakeholders have changed their minds. If the project evolves to the shrinking region, and the requirements engineering is working correctly, there is evidence that the customers are cutting down the project. This can be the indicator of a severe cut in the budget. Finally, any involution to a previous region should be considered as evidence of threats. In such cases a detailed analysis is required to assess the causes of the anomaly. This set of metrics can be collected automatically from the hypergraph and can give early alerts of threats.

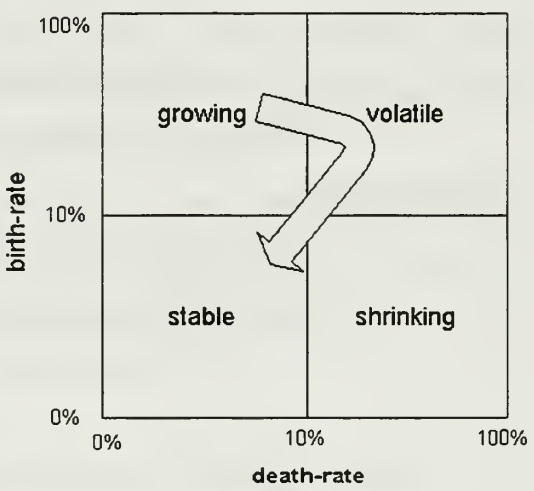


Figure 5.1: Evolution of Requirements

2. Metrics for Efficiency

The efficiency of the organization can be measured observing the fit between people and their roles in the software process. The skill match between the person and the job is required to estimate the speed in processing information and the rate of exceptions, which in turn affect the efficiency. The number of people and the turnover affect the

efficiency as consequence of productivity losses due to training, learning curves and communications. Grady and Caswell at HP identified communications complexity as one of the factors that contributes to a decrease in productivity. As the communications graph becomes more complex, that is, when it has more nodes (size), or more arcs (communication density), the information processing capacity focused on the main task decreases.

Productivity is defined as the rate of output per unit. The concept is used especially in measuring capital growth, and assessing the effective use of labor, materials and equipment. Usually, in software engineering, productivity is measured in terms of lines of code per man month. It is nonsense to use lines of code as a measure for output because only a subset of the produced lines of code is really part of the final product. Some lines of code are discarded as a consequence of the equivocality in the process; hence they do not contribute to the value of the product. The economic meaning of productivity refers to the benefit delivered. Producing large, useless software is valueless.

Fenton and Pfleeger suggest that productivity can be measured in terms of function points implemented per man month (Fenton & Pfleeger, 1997). This argument has the same drawback as in the case of lines of code. It is a fallacy to assume that function points could represent value. Function points express functionality, that is an abstract form for size, but in any case the metric was designed to provide information about value. Productivity depends on many factors like team structure, experience, and tools.

Fenton and Pfleeger suggest measuring personnel experience by an orthogonal classification of five levels (Fenton & Pfleeger, 1997. pp 419). This schema is misleading because the assessed experience of the personnel (even if the best expert did the assessment) does not provide information about the real effect of the experience on the particular project under study. The fact that a programmer was an expert does not assure that the programmer will perform as an expert. There is a second issue about this method.

The experience of the team cannot be easily derived from the experience of each individual. Human interrelations, leadership, motivation introduce a huge variability that is not considered in Fenton's approach.

Another factor affecting the productivity consists of tools and methodologies. Usually classification schemes are recommended to measure this factor (Fenton & Pfleeger, 1997), (COCOMO), (COCOMO II). Again, the measure of the kind of tool and how generalized their use is, does not provide information about the net effect on the project.

Research conducted by Brooks and Weinberg showed that team organization and dynamics contribute more to productivity than tools and methodologies (Fenton & Pfleeger, 1997). Measuring personnel productivity is difficult because as soon as people realize that they are being measured, they become resentful and the measured data could be misleading. However, a direct measure of the use of time provides objective information, is simpler to collect, and does not have the drawbacks and confounding effects explained previously. The information can be discretely collected from the scheduling mechanism (Harn, 1999f). The simulations showed that there exists an easier way to measure the productivity fit by observing the ratio between direct working time and idle. The efficiency is a measure of this fit, which is related to two risk factors: the resources and the process (see Fig. 3.2).

3. Metrics for Complexity

Complexity has a direct impact on quality because the likelihood that a component fails is directly related to its complexity. The quality of the product can only be determined at the end of the process. Hence, it is important to measure the complexity as a predictor (Munson, 1995). Real-time systems present special difficulties in terms of requirement engineering. Some requirements are difficult for the user to provide and are difficult for the analysts to determine. The best way to discover these hidden requirements

is via prototyping. CAPS is a CASE tool specially suited for this task, which uses the prototype system description language (PSDL).

The prototyping process consists of prototype construction and modification (evolution) based on evolving requirements and code generation. Both construction and modification are exploratory activities with a common target: to satisfy multiple users with different and often conflicting points of view. Requirement engineering is a consensus-driven activity in which mechanisms for conflict resolution and traceability of requirement evolution represent critical success factors.

The specifications written in PSDL can be analyzed to compute their complexity. The PSDL code has the following tokens: types, operators, data streams and constraints. Types are declarations of abstract data types required for the system. Operators and data streams are the components of a dataflow graph. Finally, constraints represent the real-time constraints that the system must support.

Two complexity metrics were defined for PSDL: **Fine Granularity Complexity Metric (FGC)**, and **Large Granularity Complexity Metric (LGC)**. The reason to compute different metrics is that they indicate two classes of threats. First, being aware of operators that are too complex is necessary. High complexity on one operator could be caused by poor design, but could be solved by further decomposition. Second, a metric that computes the total complexity of the system is required.

FGC expresses the relational complexity of each operator in the system and is a function of the fan-in and fan-out data streams related to the operator.

$$FGC = \text{fan-in} + \text{fan-out} \qquad \text{[Eq. 5-5]}$$

LGC expresses the relational complexity of the system as a function of the number of operators (O), data streams (D), and types (T).

$$LGC = O + D + T \qquad \text{[Eq. 5-6]}$$

In order to take account of all the relational complexity, LGC must be calculated by using a flattened hierarchy that contains only leaf nodes.

To analyze the PSDL code, it was necessary to develop a tool to compute the LGC and FGC. In Figure 5.2, LGC is presented under the title of "Complexity" and FGC is presented under the title "Fan-In+Fan-Out".

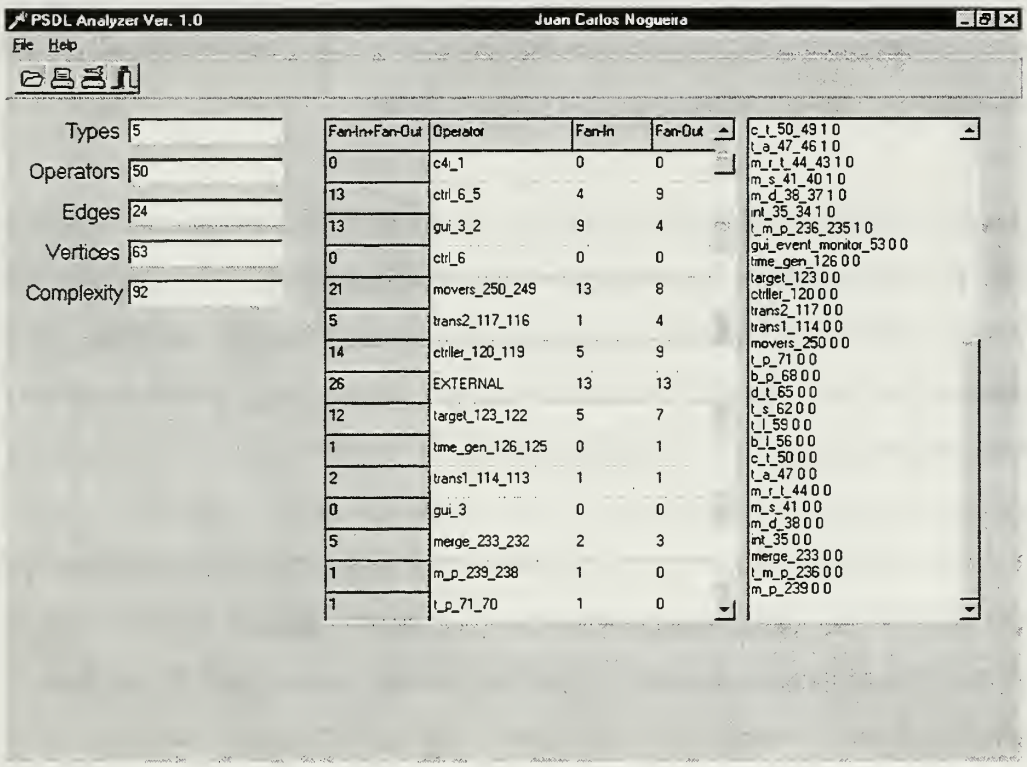


Figure 5.2: PSDL Complexity Tool

Figure 5.3 shows the strong correlation between PSDL lines of code and LGC. The correlation coefficient (R) is 0.996.

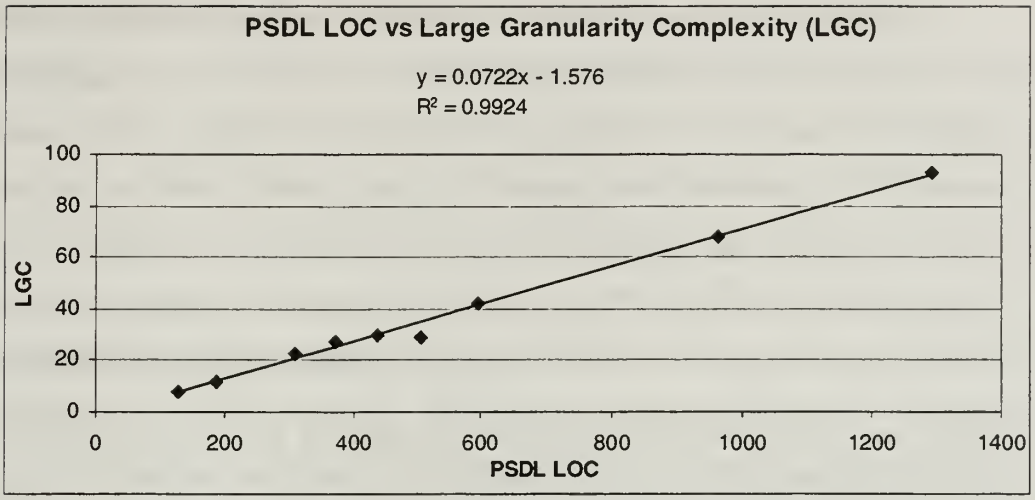


Figure 5.3: Correlation between PSDL and LGC

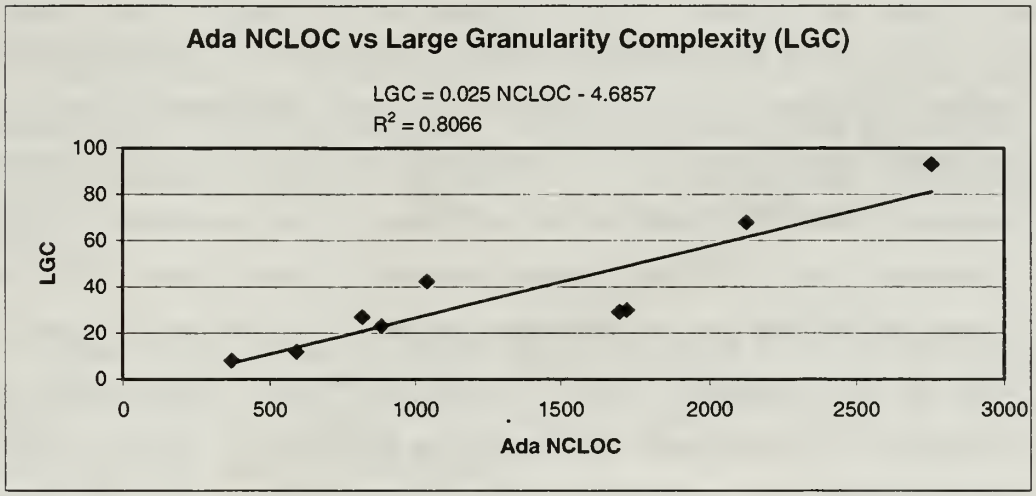


Figure 5.4: Correlation between Ada Code and LGC

The comparison between Ada non-comment lines of code of the projects with their complexity measured using LGC also shows a strong correlation (R = 0.898). The complexity metric correlates better with PSDL than with Ada. The reason for this difference is because CAPS automatically generate PSDL. On the other hand, even if CAPS generates part of the Ada code, the designer can add and modify the generated code, introducing more variability. Figure 5.4 shows the correlation observed for the same set of projects. The sample observed shows that each LGC corresponds to 40 non-comment Ada lines of code.

A caveat of this study is that the sample is small and contains only small projects, but it includes all the available information at the current time. However, the study suggests the possibility of estimating size in terms of complexity with a useful degree of accuracy. The experiment was repeated for a large project developed by the Uruguayan Navy.[21] The use of this project in the research introduced two issues. First, the project was not developed using CAPS. Second, the development language was Pascal not Ada. The first issue was easy to solve because LGC can be applied to any design that can be expressed in terms of operators, data streams, and types. Hence, by counting the number of procedures and functions, the invocations of procedures and functions, and types and units, it was possible to calculate the LGC.

The second issue imposed a different approach. Even if Ada and Pascal are third generation languages with strong typing and structure, the relationship between functional complexity (expressed in function points) and lines of code is slightly different for each language. Capers Jones suggested that each function point corresponds to 71 lines of code in the case of Ada, or 91 lines of code in the case of Pascal (Jones, 1997). However, our observations showed that the relationship between LGC and lines of code is almost the same for both languages. Table 5.1 shows the values obtained using real projects developed in Ada and Pascal. We suspect that the reason for finding a similar value for both languages is the consequence of the differences between LGC and Function Points. File management and queries, which are part of the Function Points parameters, are not considered in LGC. And in terms of operators, data streams and types both languages are quite similar.

[21] The project is a simulator developed for war gaming (SIMTAS) consisting of 75,240 lines of code in Pascal.

Table 5.1: Relationships between LGC and LOC for Ada and Pascal

Language	LOC per LGC
Ada	40
Pascal	41

B. ESTIMATION METHODS

Software projects could be considered as experiments whose cost and schedule are the output measures. It is well known that software projects tend to overrun costs and schedule (this fact has been proved by research and industry (Boehm, 1981), (Putnam, 1997), (Jones, 1996)). There are two possible ways to interpret the result of the experiment. One hypothesis is that this behavior is abnormal and is the consequence of a lack of process maturity (SEI/CMM approach). Another hypothesis is that this could be a "false-abnormal" behavior, assumed abnormal as a consequence of inappropriate measurements.

To estimate effort and time, the industry has been using three classes of tools that can be applied at different moments during the life cycle, each category being more precise than the previous one, but arriving later:

- *Very Early Estimations*. This category includes very crude approximations made during the beginning of the process usually by subjectively comparing previous projects.
- *Macro Models*. This category includes Basic COCOMO, Putnam, Function Points, etc. The estimation is done after completing the requirements phase.
- *Micro Models*. This category includes intermediate and detailed COCOMO, and Pert/CPM/Gantt techniques. The estimation is made after the design when it is possible to have a work-breakdown structure. The project estimate is the integration of all module estimates.

None of these techniques considers the following characteristics of software projects: a) requirements stability, b) personnel stability, and c) time consumed by communications, exceptions and noise in the process. All the methods use size as an input parameter as some kind of derivation from complexity. In many cases the methods to compute such complexities and sizes are questionable. Recently, Stanford University (Levitt, 1999) developed a new generation micro-model estimation tool (VitéProject) that addresses some of the previous concerns. This tool is useful to control the project, but its results arrive too late for early estimation.

How to create a macro model that considers the above concerns and can be used during the early stages of the process is an important question. Probabilities can be applied to solve this problem as detailed below. In 1939 the Swedish physicist, Waloddi Weibull, introduced a tailed probability distribution to represent the distribution of the breaking strength of materials (Devore, 1995).[22] Weibull used this distribution to model strength of Bofors's steel, fiber strength of Indian cotton, length of syrtoideas, fatigue life of steel, statures of adults males, and breadth of beans. The Weibull distribution includes the exponential and the Rayleigh as special cases. It has been used to model different failure rates: a) decreasing (when the shape parameter $\alpha < 1$), b) constant (when $\alpha = 1$ -- the exponential case with $\lambda = 1/\beta$--), and c) increasing (when $\alpha > 1$). Many authors (Johnson, 1994), (Devore, 1995) and (Lyu, 1995) advocated the use of this distribution in reliability and quality control. Others like Putnam and Norden used it to model software life cycles (Putnam, 1997). These previous works cited in Chapter II motivated the interest in this distribution.

In some literature (Devore, 1995) and software (Excel), the distribution function is presented with two parameters: α (the shape parameter), and β (the scale parameter that

[22] There is some controversy about who was the first scientist that introduced this distribution. There is a previous study of 1933 describing the "laws governing the fineness of powdered coal" that used a similar function (Johnson, 1994). Weibull distribution is also known as Weibull-Gnedenko in the Russian literature, and as Frechét for an earlier paper presented in Poland in 1927.

can compress or elongate the curve in the x axis). However, Weibull in his original work mentioned a third parameter, γ, to shift the curve to the right.

A random variable x is said to have a Weibull distribution with parameters α and β (with $\alpha > 0$, $\beta > 0$) if the probability distribution function (pdf) and cumulative distribution function (cdf) of x are respectively:

$$\text{pdf: } f(x; \alpha, \beta) = \begin{cases} 0, & x < 0 \\ (\alpha/\beta^{\alpha})\, x^{\alpha-1} \exp(-(x/\beta)^{\alpha}), & x \geq 0 \end{cases} \quad \text{[Eq. 5-7]}$$

$$\text{cdf: } F(x; \alpha, \beta) = \begin{cases} 0, & x < 0 \\ 1 - \exp(-(x / \beta)^{\alpha}), & x \geq 0 \end{cases} \quad \text{[Eq. 5-8]}$$

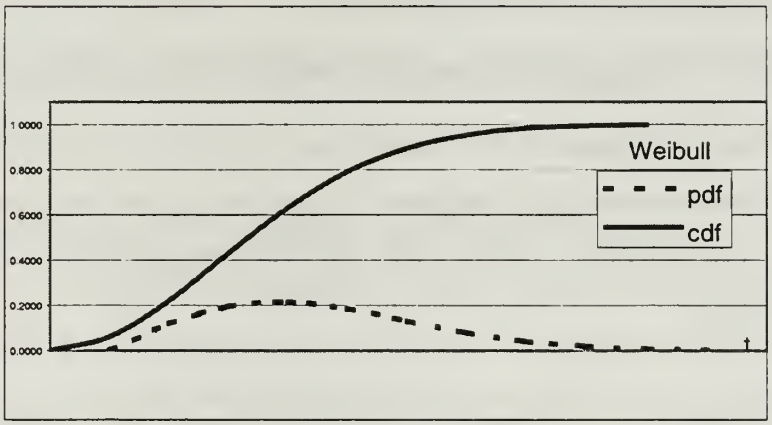


Figure 5.5: Weibull Distribution

Let's discuss the meaning of each of the variables in the function:

a) **x** is the random variable under study. In our context, x can be interpreted as development time.

b) α is a shape parameter. It affects the skew of the function. When $\alpha = 1$, the function reduces to the exponential distribution. The combined effect of α and β controls the variability of the pdf.

c) β is a scale parameter that stretches or compresses the graph in the *x* direction.

d) Note that the functions start at x = 0. A third parameter is required to shift the curves to a different starting point. For that reason, a location parameter γ was introduced, which is a function of the system's complexity. The new functions are then:

$$\text{pdf: } f(x;\gamma,\alpha,\beta) = \begin{cases} 0, & x < \gamma \\ (\alpha/\beta^{\alpha})\,(x-\gamma)^{\alpha-1}\exp(-((x-\gamma)/\beta)^{\alpha}), & x \geq \gamma \end{cases} \qquad \text{[Eq. 5-9]}$$

$$\text{cdf: } F(x;\gamma,\alpha,\beta) = \begin{cases} 0, & x < \gamma \\ 1 - \exp((-(x-\gamma)/\beta)^{\alpha}, & x \geq \gamma \end{cases} \qquad \text{[Eq. 5-10]}$$

C. CONSTRUCTION OF THE MODEL AND SIMULATIONS

1. Finding the Complexity Metric and Its Conversion to KLOC

One of the goals of this research was to provide a way to assess the duration of the project given some indicators collected during the requirements phase. In such conditions, code is not available, so the only possible measurements should come from the specification.

Research on Function Points (FP) (Albrecht 1979, 1983) showed that a clear relation between complexity and size in terms of lines of code exists. However, FP is not well-suited for real time systems or object-oriented developments. The reason is that parameters used in FP are not representative of the complexity in such systems. Chapter II discussed this issue in detail. Consequently, it was necessary to look for another way to measure complexity. The observed properties on PSDL showed characteristics that could be used to find the way to calculate complexity. In order to measure the complexity of a module, the count of the fan-in and fan-out is a good estimator. This metric was called Fine Granularity Complexity (FGC). In order to find the complexity of the whole system,

the count of PSDL operators (bubbles), data streams (arrows), and types is a good estimator. This metric was called Large Granularity Complexity (LGC).

The observations showed a strong linear correlation between the LGC and the size of the specification ($R^2 = 0.9924$). More interesting was also finding a strong (but lower, $R^2 = 0.8066$) correlation between the LGC and the size of the projects in Ada non-comment-lines of code. The size of the project in thousands of non-comment lines of code can be estimated as:

$$KLOC = (40\ LGC + 150) / 1000 \qquad \text{[Eq. 5-11]}$$

This equation was derived from the linear regression presented on Fig. 5.4.

As the complexity grows, the ratio trends to approximately 40 LOC for each unit of LGC. This finding provided us with a method to compute the size of the projects given an early measure of their complexity. This conversion is required to compare how close this approach is with respect to other methods, such as Putnam's and Boehm's, which require size as parameter.

2. Comparison between Putnam's and Boehm's Estimations

Before trying to compare this estimation model with the industry's standards (Putnam and COCOMO), an experiment was conducted to compare these two methods (see Chapter II). The experiment used Basic COCOMO because it is the only model in the family that is a macro model. Intermediate and Detailed COCOMO require a micro calibration that cannot be made until the design is done, and require subjective inputs. The purpose was to analyze early estimations without any subjectivity, so Basic COCOMO was the choice. For the comparison Putnam's results were transformed from man-years to man-months, and from years to months.

The experiment consisted of computing Basic COCOMO and Putnam for fictitious projects from 10 to 1000 KLOC. Basic COCOMO was computed for organic, semi-detached and embedded systems to discriminate between these types of projects.

The results showed that in terms of effort, Putnam's method provides an estimation that is close to the average between embedded and semidetached basic COCOMO. In terms of development time, the models are quite similar, Putnam's being more optimistic.

3. Search for the Relationship between Complexity (LGC) and Development Time

Having found a complexity metric suited for this research, the next step was to find some relationship between the LGC and development time. A simple experiment was conducted using the conversion ratio (Eq. 5-10) to obtain the size inputs for the sample. The sample points were from 1000 LGC to 30000 LGC, which means sample projects from 32 KLOC to almost 1MLOC. The average estimation for the development time using Basic COCOMO and Putnam was computed for these projects. The sample points are plotted with a smoothing thick lines (Fig. 5.6). The logarithmic function (Eq. 5-12) is plotted as a thick red line. This function has a strong logarithmic correlation (R^2 = 0.9699) with the average of COCOMO and Putnam estimations.

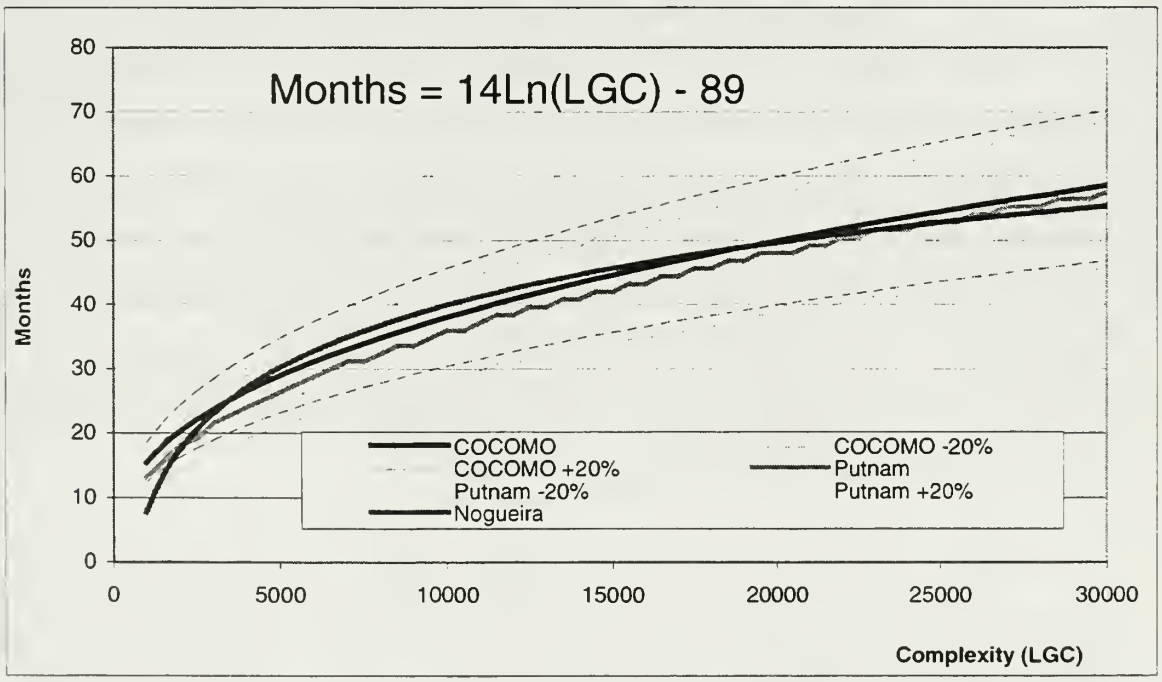


Figure 5.6: Correlation between Development Time and Complexity

Figure 5.6 shows as dotted lines the ±20% tolerances for COCOMO and Putnam.[23]

$$\text{Time (months)} = 14\ \text{Ln(LGC)} - 89 \qquad \text{[Eq. 5-12]}$$

$$\text{Time (days)} = 22 * (14\ \text{Ln(LGC)} - 89) \qquad \text{[Eq. 5-13]}$$

This equation gives a good estimation for projects between 4,000 and 20,000 LGC (128 and 640 KLOC of Ada). The estimation seems to be too optimistic for projects smaller than 1000 LGC but it is quite good for larger projects. To verify the model we used a real project consisting of 1836 LGC developed in 1.5 years by the Uruguayan Navy.[24] Equation 5-12 predicts 17 months instead of 18.

4. Search for the Relation between Efficiency and Development Time

By applying causal analysis, it was found (Chapter III) that the risk of the project should depend on three factors: complexity, productivity and volatility of requirements. The previous sections concerned a method to compute complexity, as well as an equation to estimate the development time (in months), based on complexity (Eq. 5-12).

Productivity is classified into four categories by time spent at work:

- *Direct.* Time spent working and correcting errors on the product. In VitéProject terminology, it is the sum of work and rework.
- *Indirect.* Time spent in activities supporting the work, such as meetings, coordination, information exchanges, etc. In VitéProject terminology, it is known as coordination time.
- *Idle.* Time spent without work to do, waiting for some input. In VitéProject terminology, this is known as waiting time.

---

[23] Boehm says that BASIC COCOMO can only predict within the ±20% on 25% of the cases (Boehm, 1981 pp. 495).

[24] SIMTAS a simulator for war gaming with 75,240 lines of code.

- *Personal.* Time spent doing anything except the other categories. VitéProject does not compute this category of time. However, it is loosely related to the noise parameter of the tool.

Examining the time distribution of these categories, a remarkable pattern appeared differentiating the high efficiency scenarios from the low efficiency ones. This effect was independent of the other two variables of the simulation. Hence, this suggested that the time distribution could be a good indicator for the efficiency of the organization The ratio between work and idle time can be automatically captured from the software evolution steps as suggested by (Ham, 1999f).

Figure 5.7 presents the average distribution times for the simulated scenarios. A pattern of time distributions can be clearly observed. Scenarios with low productivity have a percentage of idle time greater than 13% of the total development time. The following characteristics can be observed from the simulations:

- Direct work is reduced by 10% when the efficiency is high.
- Indirect work is reduced by 40% when the efficiency is high.
- Idle time is reduced by 70% when the efficiency is high.

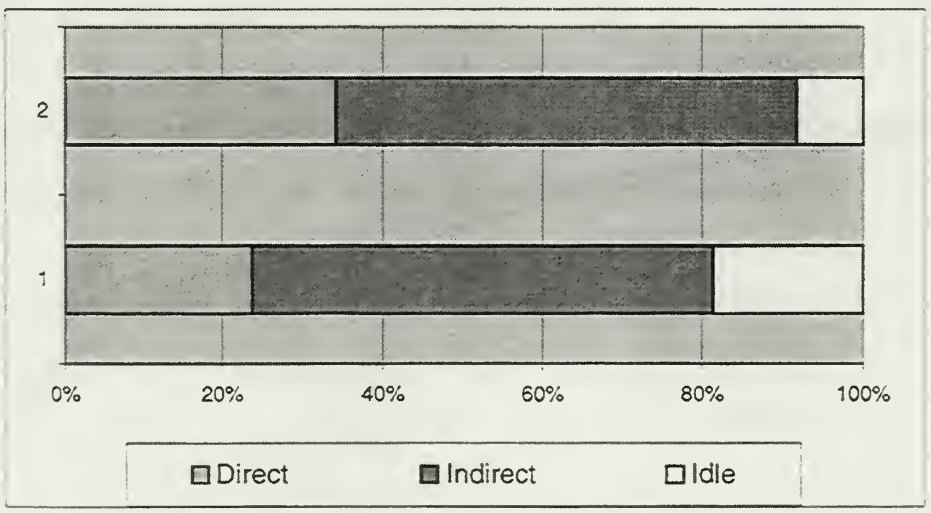


Figure 5.7: Patterns of Time Distribution. The bar number 1 shows the time distribution for a low efficiency scenario. The bar number 2 shows the time distribution for a high efficiency scenario.

Low and high efficiency scenarios can be differentiated by the ratio of the percentage of direct time over percentage of idle time, which was called efficiency ratio (EF):

$$EF = Direct\% / Idle\% \qquad [Eq. 5\text{-}14]$$

For high efficiency scenarios $2.0 < EF$, and for low efficiency scenarios $0.8 < EF < 2.0$. The simulations showed that for high efficiency scenarios the development time was sensibly shorter than for low efficiency ones. The reasons the ratio EF is related to productivity require further study. However, one can conjecture that the reason could be the wasted time waiting for other's work. This inefficiency can also be related to:

- The mismatches between roles and people skills.
- People turnover, generating noise and productivity losses derived from training and learning curves.
- Number of people, influencing the productivity in two ways. If the number of people is less than the roles of the software process, then the productivity will be affected because someone's attention and effort will be divided into more than one role. On the other hand if the number of people exceeds the roles, then the productivity will be affected by additional communications.

5. Configuration of VitéProject for the Simulation[25]

To calibrate the values of the parameters described previously, a set of simulations with VitéProject was conducted keeping the values of two variables constant and changing the third one. This is done to isolate the effects of each variable. For efficiency and requirements volatility two values were used: L for low and H for high. The

[25] The details about VitéProject are discussed in Chapter II, Section G.

complexity received four different values so that the effect of complexity could be observed in detail.[26].

Having three variables and using two possible values for two of them and four values for the third one, the universe of scenarios consists of the sixteen ($4 * 2^2$) scenarios showed in Table 5.2.

Table 5.2: Simulated Scenarios

Scenario name	*EF*	*RV*	*CX*	*Experience Adjust.*	*Requir. Adjust.*	*Work FTE factor*	*Complex. Adjust.*
LLL	Low	Low	Low	Low	Low	1	Low
LLH	Low	Low	High	Low	Low	1	High
LLH2.5	Low	Low	High 2.5	Low	Low	2.5	High
LLH5	Low	Low	High 5	Low	Low	5	High
LHL	Low	High	Low	Low	High	1	Low
LHH	Low	High	High	Low	High	1	High
LHH2.5	Low	High	High 2.5	Low	High	2.5	High
LHH5	Low	High	High 5	Low	High	5	High
HLL	High	Low	Low	High	Low	1	Low
HLH	High	Low	High	High	Low	1	High
HLH2.5	High	Low	High 2.5	High	Low	2.5	High
HLH5	High	Low	High 5	High	Low	5	High
HHL	High	High	Low	High	High	1	Low
HHH	High	High	High	High	High	1	High
HHH2.5	High	High	High 2.5	High	High	2.5	High
HHH5	High	High	High 5	High	High	5	High

The first column contains the name of the scenario expressed as the initials of the values of the three input parameters of the model: efficiency (EF), requirements volatility (RV), and complexity (CX). The next three columns show the values of the three input parameters. The next four columns show the parameters used in VitéProject to model each scenario (Levitt, 2000). The inputs for the simulation are:

[26] The details about how to express complexity in VitéProject is discussed in Chapter II, Section G.3.

(a) *Efficiency*. Efficiency is expressed in terms of the following parameters of VitéProject: *Team Experience*, *Application Experience*, and *Skill Levels*. For the low efficiency scenarios all these three parameters are set to "low." For high efficiency scenarios the values of these three parameters are set to "high."

(b) *Requirements Volatility*. Requirements Volatility is expressed in terms of two parameters of VitéProject that express the difficulty in the elucidating the requirements: *Uncertainty*, and *Requirement Complexity*. The values of these two parameters are set to "high" or "low" depending on the requirement volatility of each scenario.

(c) *Complexity*. The complexity is expressed in two ways. First, by using the parameter of VitéProject called *Solution Complexity*. This parameter is set to "low" in the scenarios with low complexity level (L). And it is set to "high" for the other complexity levels (H, H2.5, and H5). Second, the complexity is expressed in terms of expected time for each activity, as in PERT, Gantt, or CPM. The values of expected time for each activity are calculated using the estimated time for the project and the work breakdown structure. The complexity levels L and H correspond to 781 LGC. However, applying the value "low" to the parameter Solution Complexity, the result is a decrease in the total complexity to 746 LCG. The complexity values for scenarios with complexity levels H2.5 and H5 correspond to the estimated development times for the project considering that the time for each activity is increased by a factor 2.5 and 5 respectively. These values correspond to 1334 LGC and 3230 LGC respectively. These four values of LGC are used in all the scenarios (see Table 5.3). The differences in the result of the column E(t) obey to the changes in the parameters *Efficiency* and *Requirements Volatility*.

VitéProject parameters of the organization were configured as follows:

- *Formalization*: Formalization is the degree of informal communications among actors. This parameter was set to "low" to indicate likely informal communications.
- *Centralization*: Centralization is a qualitative degree of decision making and exception handling on lower levels. This parameter was set to "low" to indicate that lower levels have decision making power.
- *Matrix Strength*: This parameter was set to "high" to indicate that it is non-departmental work.
- *Team Experience*: This parameter was set to "low" or "high" depending on the efficiency simulated in each particular scenario.

The simulation tool was configured to run 30 simulations for each scenario, and the organizational parameters were set to match the characteristics of software development. Appendix C provides further details about the parameterization of VitéProject. These values are the consequence of an analysis achieved with OrgCon (Obel & Burton, 1998). The characteristics of a fictive software development organization (CMM level 2-3) was introduced in the simulations. The reports of the expert system are presented in Appendix A.

Table 5.3 shows the expected durations and the standard deviations in days for the sixteen scenarios simulated. The output of the simulation are the estimated development time (E(t) days) and the standard deviation (SD(t) days). The tool also provides the CPM estimate of the development time. The column titled LGC shows the complexity measure for each scenario measured in LGC. Table 5.2 provides also the estimated duration in months and days computed using Eq. 5-13. The samples used in the research cover projects from two to 26 months (2.4 years). This range of projects was selected because the rapid changes in technology introduce variations that can distort any possible estimate. For hardware the technology horizon is 18 months (Intel). For software is less than two years (Microsoft).

Table 5.3: Simulation Results

Scenario	EF	RV	CX	E(t) days	SD(t) days	CPM (days)	LGC	Months	days
LLL	L	L	L	88	5	67	746	3.61	79
LLH	L	L	H	101	6	67	781	4.25	93
LLH2.5	L	L	H2.5	254	16	168	1334	11.74	258
LLH5	L	L	H5	507	31	335	3230	24.12	531
LHL	L	H	L	101	7	67	746	3.61	79
LHH	L	H	H	128	10	67	781	4.25	93
LH2.5	L	H	H2.5	319	25	168	1334	11.74	258
LH5	L	H	H5	638	49	335	3230	24.12	531
HLL	H	L	L	32	2	67	746	3.61	79
HLH	H	L	H	42	3	67	781	4.25	93
HLH2.5	H	L	H2.5	105	7	168	1334	11.74	258
HLH5	H	L	H5	209	14	335	3230	24.12	531
HHL	H	H	L	42	3	67	746	3.61	79
HHH	H	H	H	49	4	67	781	4.25	93
HHH2.5	H	H	H2.5	122	9	168	1334	11.74	258
HHH5	H	H	H5	244	18	335	3230	24.12	531

VitéProject provides an estimate very close to the value calculated from Eq. 5-13 for all the scenarios where the efficiency and the requirements volatility are low. The simulation results are more pessimistic when the requirements volatility is high. As expected, VitéProject estimates are different from the CPM estimates. The difference is due to the communication and exception handling effects. Observe that VitéProject provides a more conservative estimate than CPM, for the scenarios with low efficiency. Equation 5-13 is more conservative than the CPM estimate for all the cases.

The simulation data can be found in Appendix B. To analyze the effect of efficiency, the results of the simulations scenarios Lxx were compared against scenarios Hxx. To analyze the effect of requirement volatility, the results of the simulations of scenarios xLx were compared against scenarios xHx. To analyze the effect of complexity, the results of the simulations of the following scenarios were compared: xxL vs xxH, xxH2.5, and xxH5.

As explained before, VitéProject was configured as an organization at CMM 2-3. To verify behavior of the model under other CMM levels, we configured the tool for CMM 1 and CMM 5 by changing the organizational parameters according to the

recommendations of OrgCon (see Appendix B). OrgCon recognized low formalization and high centralization for CMM 1, and high formalization and medium centralization for CMM 5. The changes in the parameters *formalization* and *centralization* made little difference in VitéProject. The simulation result shows that there is no statistical difference for the development times at different CMM levels. This result was surprising because we should expect a better performance at CMM 5. The reason for this result could be a limitation on the simulation capability of VitéProject to address characteristics of CMM 5. The tool has been used with success at software organizations CMM 2-3 only. ANOVA at level 0.05 shows that the changes in the organizational design have very little impact on the result. Statistically, the samples CMM 2-3 cannot be differentiated from the samples CMM 1 or 5. The P-values are higher than 0.05, hence the hypothesis of same population cannot be rejected at level 0.05.

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Level 2-3	16	2980.8	186.3	30401.12
Level5	16	3729	233.0625	57892.06

ANOVA level 0.05

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	17493.85	1	17493.85	0.396267	0.533788	4.170886
Within Groups	1324398	30	44146.59			
Total	1341892	31				

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Level 2-3	16	2980.8	186.3	30401.12
Level1	16	3675	229.6875	48056.6

ANOVA level 0.05

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	15059.8	1	15059.8	0.383896	0.540203	4.170886
Within Groups	1176866	30	39228.86			
Total	1191926	31				

VI. STATISTICAL ANALYSIS OF THE OBSERVED RESULTS AND DERIVATION OF THE MODELS

A. SUMMARY OF THE OBSERVED RESULTS

The simulations conducted showed that the three risk factors observed during the causal analysis (efficiency, requirements volatility, and complexity) have compound effects over the three parameters of the Weibull distribution.

The samples created by simulation were analyzed with Weibull++ Ver 5.0 (ReliaSoft, 2000), a statistical software package that identifies the distribution that best fits the sample. The expected distribution was confirmed. For all the samples, the tool identified Weibull with three parameters as the best fit. Appendix D contains the statistical analysis of the samples obtained by simulation.

The shape parameters (α) were close to 2.0, but not exactly 2.0. I expected to identify Rayleigh curves, as Norden and Putnam had previously done. The origin of the difference was not clear. A preliminary suspicion about the size of the samples was discarded. A larger set of samples showed that the values remained close to 2.0 with the same standard error. The author contacted Dr. Lawrence Putnam about this issue and this was his answer:

> *That much variation is not abnormal. In fact, it is pretty normal. We saw similar variation and then assumed a value of 2.0 for implementation work. Most of the empirical results you get in software research vary a lot because the data are almost always noisy.*
>
> *Your approach sounds very interesting. I think it is imperative that updates to estimates be done all along the time line. I have not tried the approach you are taking, but I see no apparent reason it would not work and it certainly is worth investigating.*

As you have probably picked up from the books I sent on, we have based an initial estimate on our software equation calibrated to the development organization and then bounded the s/w equation with appropriate management constraints–schedule, staff, money, reliability, etc. Then, once the project is underway, we use a dynamic version of the Rayleigh equation in several dependent variable dimensions and do curve fitting and weighting based on the quality of the fits. This process converges and gives pretty good estimates by the time you are one-third of the way into the main software construction phase. The estimates continue to get better as more and more real data are fed into the process. Our tool that does this is SLIM Control.

One curious thing that we have discovered on a number of real projects where the size estimate is poor or quite uncertain and the productivity parameter is also poor or uncertain is the fact that as time-based data on staffing, defects, function completed so far is fed into the curve fitting process the estimate gets better and becomes quite reliable in predicting the end point for schedule and the associated cost and reliability. So, this is telling us that the ratio of size/productivity parameter is being determined quite accurately by the curve fit even though we don't know the numerator, or the denominator, or both, very well. This indicates to me there is still something in the process that we don't understand very well yet–even though the empirical outcome is very usable from an engineering viewpoint.

The simulations showed that the efficiency level seems to have effects over α and γ. For instance, when the efficiency level is high the mean value of α is 1.95 and it grows to 2.5 when the efficiency is low. Variations on complexity or requirements volatility do not affect the value of α. This seems to provide evidence that α is associated with the efficiency of the project.

The efficiency also affects the delay parameter γ. When the efficiency level was high, γ is approximately 28% of the time derived from LGC. When the efficiency level was low, the value of γ is approximately 76% of the time derived from LGC.

Variations in efficiency, complexity, and requirements volatility all seemed to affect β. For all the cases β was proportional to γ ($\beta \cong \gamma/5.5$ with some variations depending on the efficiency level and the requirements volatility). For 50% of volatility

the γ/β ratio decreases to 2.5. For values close to 100% of volatility the γ/β it is 1.1. When the requirements volatility is not a major issue, the value of 5.5 could be a conservative estimate.

The Two-way ANOVA shows that Complexity is the main contributor to variation in scenarios with high or low Efficiency. Complexity affects 95% for low efficiency scenarios and 97% for high efficiency ones.

ANOVA (EF = H)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample (RV)	17733.20417	1	17733.2	204.9622	9.69E-34	3.881851
Columns (CX)	1382083.513	3	460694.5	5324.754	1.4E-213	2.64351
Interaction	7242.279167	3	2414.093	27.90233	1.92E-15	2.64351
Within	20072.5	232	86.5194			
Total	1427131.496	239				

CX effect	96.84%
RV effect	1.24%
Combined effect	0.51%
Random effect	1.41%

ANOVA (EF = L)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample (RV)	207035.0042	1	207035	372.7349	3.61E-50	3.881851
Columns (CX)	8810456.713	3	2936819	5287.294	3.2E-213	2.64351
Interaction	125760.5458	3	41920.18	75.47088	4.22E-34	2.64351
Within	128864.0333	232	555.4484			
Total	9272116.296	239				

CX effect	95.02%
RV effect	2.23%
Combined effect	1.36%
Random effect	1.39%

B. THE MODELS

This section introduces a set of estimation models with increasing degree of accuracy based on:

- metrics from the three risk factors
- Weibull cumulative density function [Eq. 6-1]
- the derivation of the time [Eq 6-2]

The cdf of Weibull is:

$$P(x \leq t) = p = 1 - \exp(-(((t - \gamma) / \beta)^{\alpha})) \qquad \text{[Eq. 6-1]}$$

$$\Leftrightarrow \quad 1 - p = \exp(-(((t - \gamma) / \beta)^{\alpha}))$$

$$\Leftrightarrow \quad \ln(1 - p) = -(((t - \gamma) / \beta)^{\alpha})$$

$$\Leftrightarrow \quad -\ln(1 - p) = (((t - \gamma) / \beta)^{\alpha})$$

$$\Leftrightarrow \quad (-\ln(1 - p))^{1/\alpha} = (t - \gamma) / \beta$$

$$\Leftrightarrow \quad \beta\,(-\ln(1 - p))^{1/\alpha} = t - \gamma$$

$$\Leftrightarrow \quad t = \beta\,(-\ln(1 - p))^{1/\alpha} + \gamma \qquad \text{[Eq. 6-2]}$$

Eq. 6.2 provides the estimated time for a given probability of success p. Note that t and γ should be expressed in the same units, β is unitless. From the properties of the Weibull, we can express the expected value of t in terms of the Gamma function as follows:

$$E(t) = \beta\Gamma(1 + 1/\alpha) + \gamma \qquad \text{[Eq. 6-3]}$$

The following notation applies to the algorithms that define the models:

EF: efficiency level as a real derived from Eq. 5-14.

RV: requirements volatility as percentage derived from Eq. 5-4.

CX: complexity in LGC (derived from Eq. 5-5).

γ: delay in days (derived from Eq. 5-13).

All the algorithms can be used to obtain t given EF, RV, CX, and p (the probability of finishing at time t or before); or to obtain p given EF, RV, CX, and t (a given day in the future).

Model 1: This model can be used when the requirements volatility is small.

Algorithm Model 1:

```
// Inputs: EF, LGC, t
// t is given in days, we assume 22 working days per month
// Output: p = P(x<=t)
If (EF > 2.0)      then  begin
                         α = 1.95;
                         γ = 22 * 0.28 * (14 * ln(LGC) - 89);
                         end
                   else  begin
                         α = 2.5;
                         γ = 22 * 0.76 * (14 * ln(LGC) - 89);
                         end;
β = γ / 5.5;
p = 1 - exp(-(((t - γ) / β)^α));      // P(x<=t)
```

Model 2: This model considers the three factors (EF, RV, and CX), but it neglects the combined effect of EF and RV.

Algorithm Model 2:

```
// Inputs: EF, RV, CX, t
// t is given in days, we assume 22 working days per month
// Output: p = P(x<=t)
If (EF > 2.0)      then  begin
                         α = 1.95;
                         γ = 22 * 0.28 * (14 * ln(LGC) - 89);
                         end
                   else  begin
                         α = 2.5;
                         γ = 22 * 0.76 * (14 * ln(LGC) - 89);
                         end;
If (RV > 30)       then  β = γ / 5.25       // RV more than 30%
                   else  β = γ / 5.9;
p = 1 - exp(-(((t - γ) / β)^α));      // P(x<=t)
```

Model 3: This model considers the three factors as well as the combined effects of EF and RV.

Algorithm Model3:

```
// Inputs: EF, RV, CX, t
// t is given in days, we assume 22 working days per month
// Output: p = P(x<=t)
If (EF > 2.0)     then  begin
                        α = 1.95;
                        γ = 22 * 0.32 * (14 * ln(LGC) - 89);
                        β = γ /(5.71 + (RV - 20) * 0.046);
                        end
                  else  begin
                        α = 2.5;
                        γ = 22 * 0.85 * (14 * ln(LGC) - 89);
                        β = γ /(5.47 - (RV - 20) * 0.114);
                        end;
p = 1 - exp(-(((t - γ) / β)^α));      // P(x<=t)
```

Models 1, 2 and 3 can be used only under the following assumptions:

$CX \geq 600$ LGC

$RV \leq 67\%$

These three models were tried against 16 simulated projects obtaining the scatter plots of Fig. 6.1, 6.2 and 6.3 respectively. These estimates were calculated using 95% of confidence ($p = 0.95$). Note the errors as vertical segments between the estimated and real values. The values of R and R^2 are shown in Table 6.1.

Table 6.1: Correlations for Models 1, 2 and 3

	Model 1	*Model 2*	*Model 3*
R	0.9867	0.9890	0.9930
R^2	0.9736	0.9781	0.9862

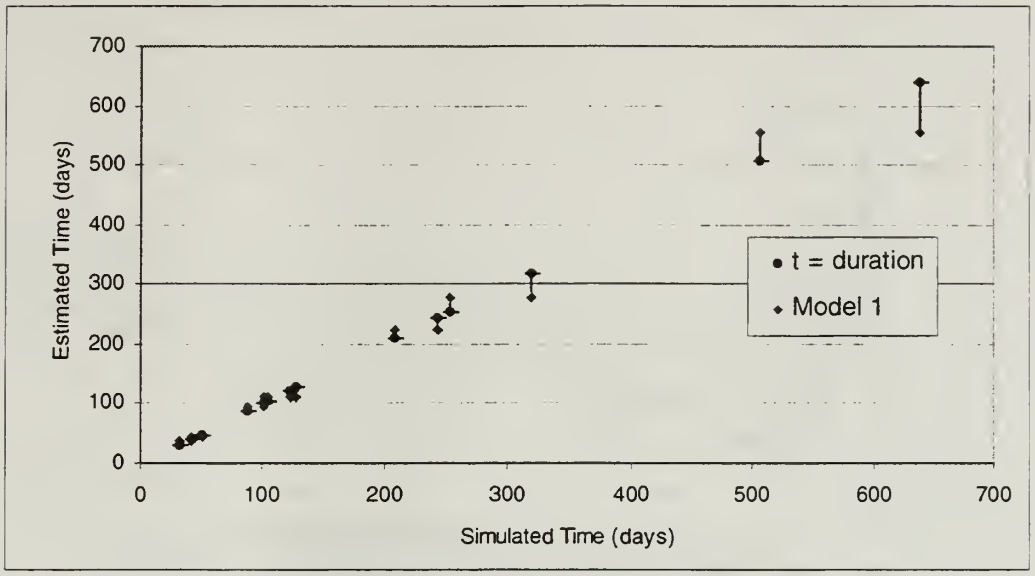


Figure 6.1: Scatter Plot of Model 1

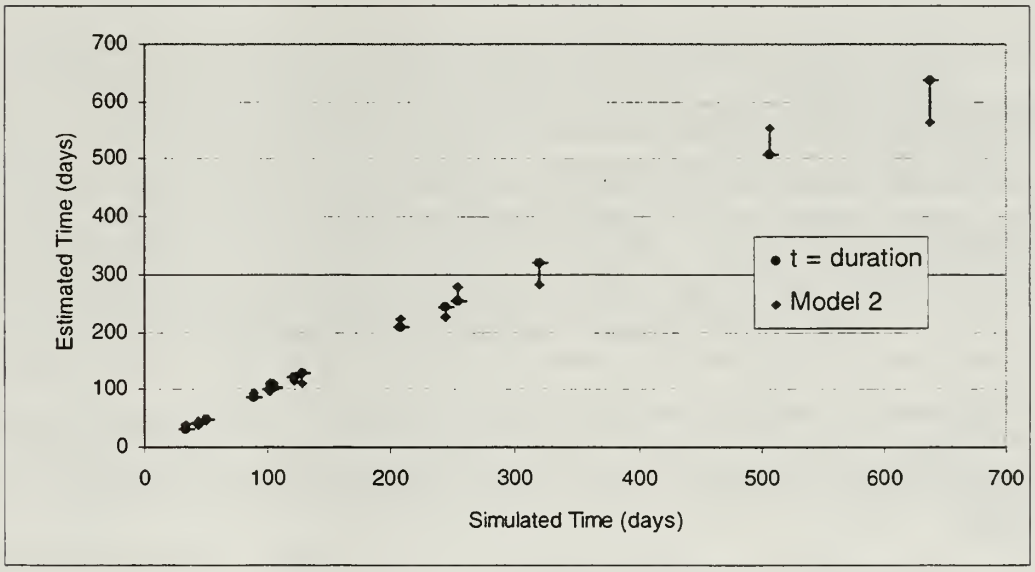


Figure 6.2: Scatter Plot of Model 2

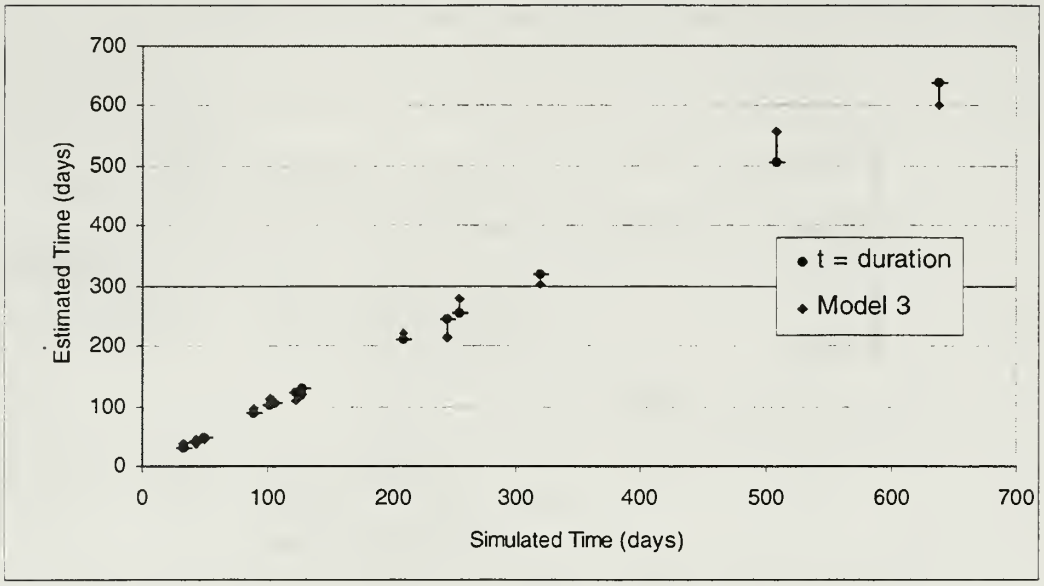


Figure 6.3: Scatter Plot of Model 3

The analysis of variance (ANOVA) shows that the samples obtained from the simulations and the samples obtained from the estimates using Model 1, 2 or 3 cannot be statistically differentiated. I tested the hypothesis Ho: "The samples obtained from the simulation and the sample obtained by the model belong to the same population" with $\alpha = 0.05$. The results of the ANOVA analysis are in Appendix D. The P-values much greater than α, show that Ho cannot be rejected. The P-value is the smallest level of significance at which Ho would be rejected. Once the P-value has been determined, the conclusion at any particular level α results from comparing the P-value to α (Devore, 1995):

1. P-value $\leq \alpha \Rightarrow$ reject Ho at level α
2. P-value $> \alpha \Rightarrow$ do not reject Ho at level α

The P-values found are 0.943641 for model 1, 0.955883 for model 2, and 0.998394 for model 3. The accuracy of the estimations can also be observed using the boxplots technique (Fig. 6.4). Observe the increasing accuracy in the outliers on the models from model 1 (the simplest) to model 3 (the most refined).

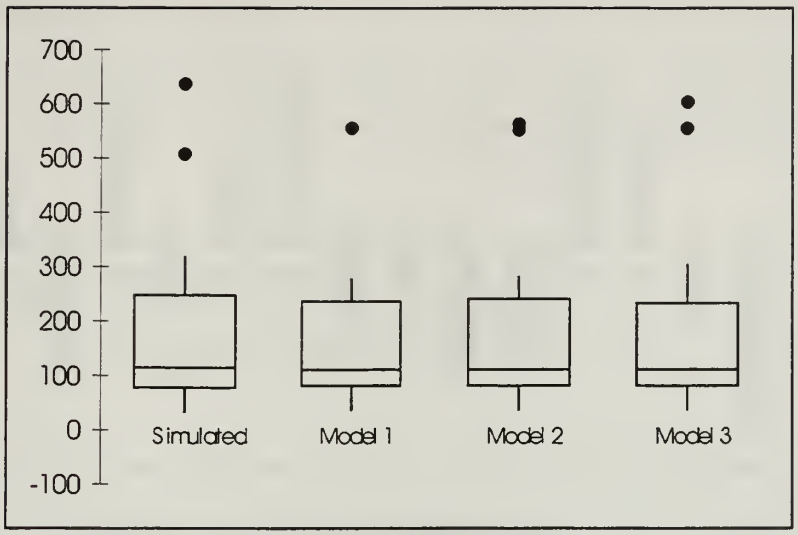


Figure 6.4: Boxplots of the Estimates from the Simulations and the Models

Another interesting result is that the errors remain in the range of ±15% for all the scenarios. This result is interesting if we compare it with the results of COCOMO (±20% in the best cases). Barry Boehm in reference to the validation of COCOMO said: "In terms of our criterion of being able to estimate within 20% of projects actuals, Basic COCOMO accomplishes this with only 25% of the time, Intermadiate COCOMO 68% of the time, and Detailed COCOMO 70% of the time" (Boehm, 1981 p. 495).

The errors seem to be stable. Figure 6.5 shows that the increase in the duration of the project has no effect over the percentage of error. The errors of my model seem to be stable and in the order of ± 15%.

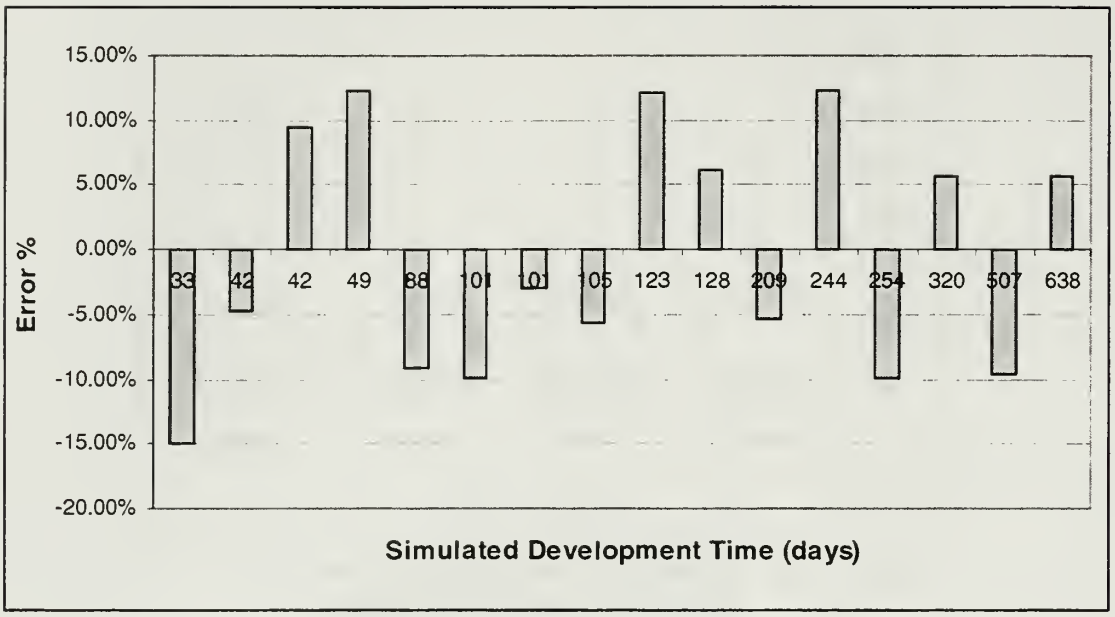


Figure 6.5: Errors for Model 3

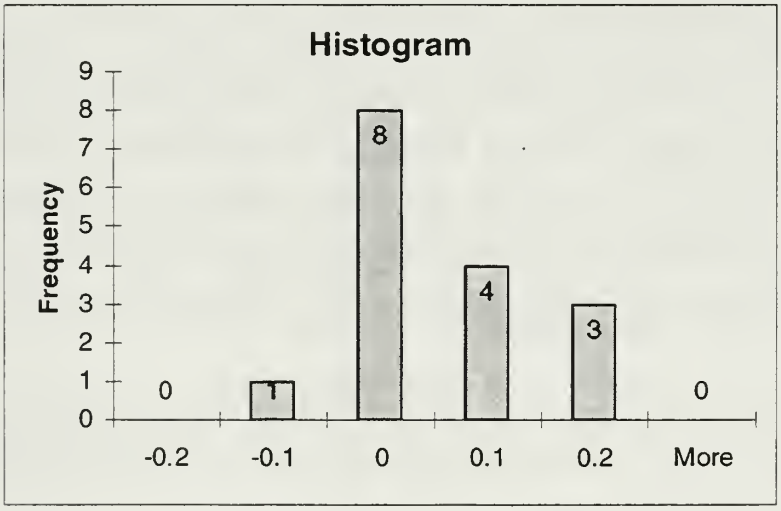


Figure 6.6: Error Histogram

Figure 6.7 shows the histogram of the errors. Even if the majority of the errors are in the zero and negative bins, that is the estimations were more conservative than the simulations, there are seven cases in which the estimates were short. However, the estimation was never wrong in more than 15%. This information should be considered for practical project management decisions.

The assumptions for models 1, 2 and 3 restrict the use of the models. For that reason a fourth model was introduced. Model 4 can be used for any range of complexity and requirements volatility. The model uses an adaptation of the general form of Eq. 5-13 as follows:

$$\text{Time (days)} = 22 * 14 \text{ Ln (LGC)} - 89$$

$$\text{Time (days)} \cong 22 * \text{Ln} (1 + \text{LGC}^{14} / \exp(89))$$

$$= 22 * \text{Ln} (1 + \text{LGC}^{a} / \exp(b)) \qquad \text{[Eq. 6-4]}$$

The values for the coefficients *a* and *b* are affected by the efficiency. The value of γ is obtained by multiplying Eq. 6-4 by an adjustment due to the requirements volatility. The ratios for the observed γ at high and low requirements volatility are almost constant (1.18) despite the efficiency level.

That is

$$\gamma_H/\gamma_L = 1.18 \qquad \text{[Eq. 6-5]}$$

The subscripts $_H$ and $_L$ represent the requirement volatility levels.

The higher requirements volatility level in the simulations corresponded to RV = 40%, and the lowest corresponded to RV = 0%. Hence form Eq. 6-5 one can derive the adjustments due to requirements volatility used in the algorithm Model 4.

Model 4: This model can be used for any range of complexity and requirements volatility, and considers the three factors, their combined effects, and the following a priori assumptions:

- A project with 0 LGC will take 0 days.
- α, β, and $\gamma > 0$.
- If RV increases then p(x<=t) decreases.
- If CX increases then p(x<=t) decreases.
- If EF increases then p(x<=t) increases.

Algorithm Model 4:

```
// Inputs: EF, RV, CX, t
// t is given in days, we assume 22 days per month
// Output: p = P(x<=t)
If (EF > 2.0) then begin
                    α = 1.95;
                    γ = 22 * ln(1 + LGC^4.5/exp(28.5))*(1 + 0.0045 * RV);
                    β = (γ/5.71);
                    end
               else begin
                    α = 2.5;
                    γ = 22 * ln(1 + LGC^12/exp(76))*(1 + 0.0045 * RV);
                    β = (γ/5.47);
                    end;
p = 1 - exp(-(((t - γ)/β)^α));
```

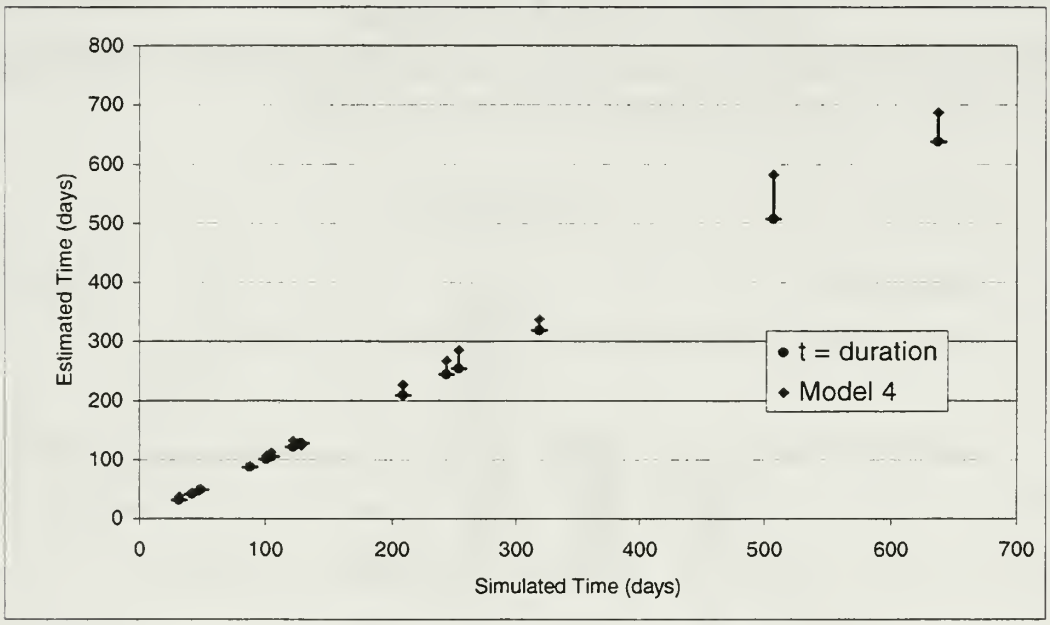


Figure 6.7: Scatter Plot for Model 4

Figure 6.7 shows the scatter plot comparing the simulated times versus the estimated times. The correlation coefficient for model 4 is R = 0.998839 (R^2 = 0.997679). Model 4 is conservative. As Fig. 6.8 shows, most part of the errors are overestimations. The errors seem to be stable, Fig. 6.8 shows that the increase in the duration of the project has no effect over the percentage of error.

The percentage of error was calculated as:

error % = (simulated time - estimated time) / estimated time

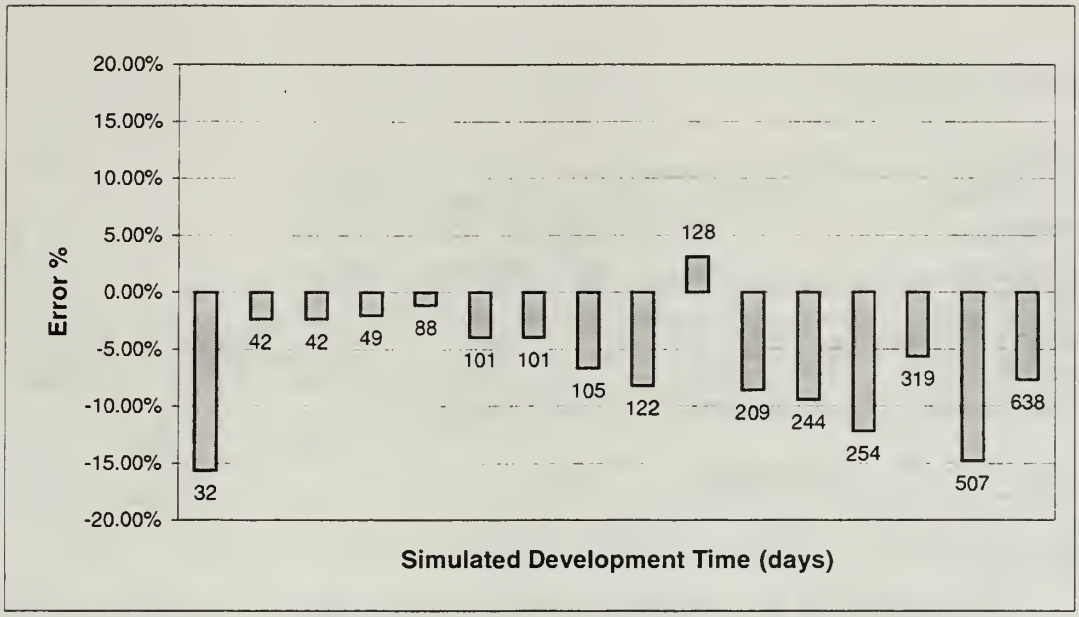


Figure 6.8: Errors for Model 4

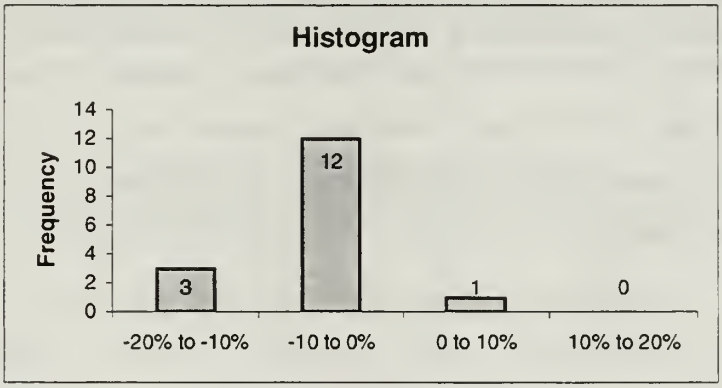


Figure 6.9: Histogram of Errors for Model 4

The negative errors in the histogram (Fig. 6.9) show that the model is conservative. The maximum overestimation error was less than 16%, and the maximum underestimation was less than 4%

The hypothesis Ho: "the samples obtained from the simulation and the samples from model 4 estimates are from the same population" cannot be rejected with level $\alpha = 0.05$. The P-value much greater than α shows that Ho cannot be rejected.

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Model 4	16	3225	201.5625	37409.86
t = duration	16	2981	186.3125	30372.1

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1860.5	1	1860.5	0.054897	0.816344	4.170886
Within Groups	1016729.38	30	33890.98			
Total	1018589.88	31				

The analysis of the simulations results revealed that the complexity and the time are related by a logarithmic function. This observation confirms our expectations discussed in Chapter V, Section C.3 (Eq. 5-13). Figure 6.10 shows the different curves for four combinations of efficiency and requirements volatility levels. The graph presents also the plot for Eq. 5-13. The early estimates obtained from Eq. 5-13 are relatively close to the results for scenarios with low efficiency and low requirements volatility. All the curves are of the form:

$$y = 22 * (a \, Ln(x) - b) \qquad \text{[Eq. 6-6]}$$

$$= 22 * Ln(1 + x^a / \exp(b)) \qquad \text{[Eq. 6-7]}$$

where x is the complexity expressed in LGC.

a, b are coefficients presented in Table 6.2.

y is the time expressed in days.

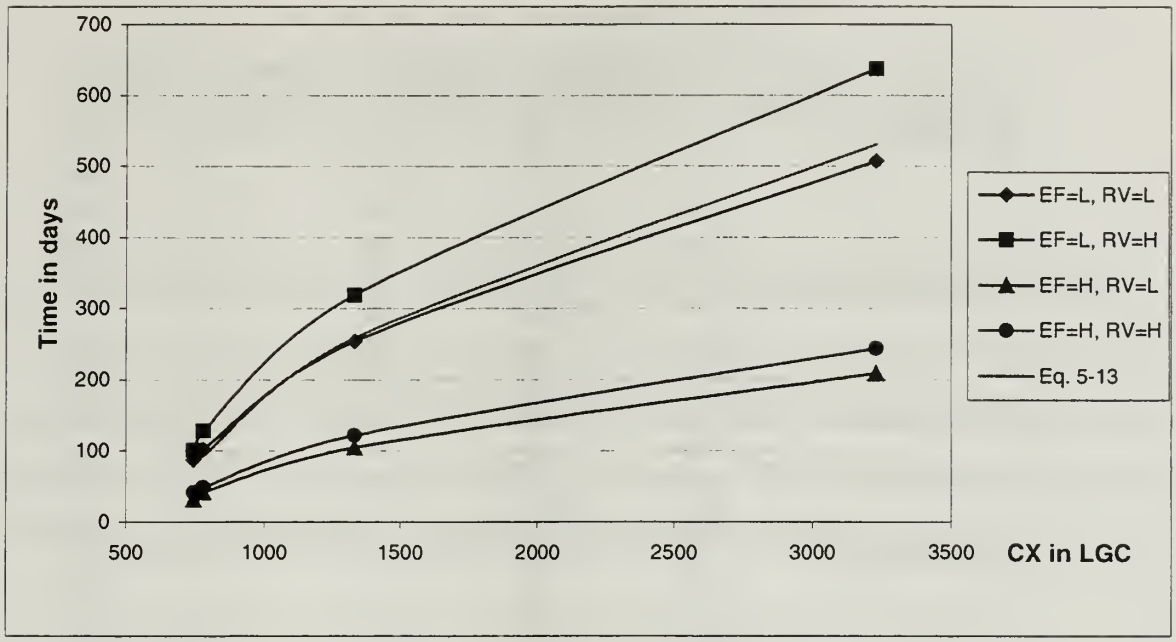


Figure 6.10: Development Time Based on the Final Complexity for Different Combinations of Efficiency and Requirements Volatility

Table 6.2: Coefficients for Eq. 6.7

Scenario	*a*	*b*
EF = L, RV = 0%	16.5	104
EF = L, RV = 40%	13	82
EF = H, RV = 0%	5.5	34.3
EF = H, RV = 40%	6.25	39.5

Equation 6-7 and Table 6.2 can be applied only when all the requirements are frozen and completely specified. As Table 6.3 shows, Eq 6-7 provides very accurate estimates but later in time. That is when the requirements are frozen and one can estimate with confidence the total complexity of the system. At that moment in time COCOMO 81 or SLIM can also be applied.

Table 6.3: Estimation Errors for Eq. 6-7

Scenarios	CX LGC	Simulated Time (days)	Model 4 Estimates	Error (days)	Error %
EF=L, RV=0%	746	88	88	0	0.00%
	781	102	101	1	0.98%
	1334	254	254	0	0.00%
	3230	507	507	0	0.00%
EF=L, RV=40%	746	101	107	-6	-5.94%
	781	128	123	5	3.91%
	1334	319	318	1	0.31%
	3230	638	639	-1	-0.16%
EF=H, RV=0%	746	32	35	-3	-9.38%
	781	42	40	2	4.76%
	1334	105	104	1	0.95%
	3230	209	209	0	0.00%
EF=H, RV=40%	746	42	42	0	0.00%
	781	49	49	0	0.00%
	1334	122	122	0	0.00%
	3230	244	244	0	0.00%

VII. INTEGRATION TO CAPS

A. INTEGRATION WITH THE EVOLUTIONARY SOFTWARE PROCESS

The evolutionary prototyping software process (Fig. 7.1) is a directed graph with two cycles. Initially, the analysts collect a set of issues, which represent the concerns and preliminary goals of the customers, and transform them into a more elaborated level of description called requirements using a requirements analysis step.

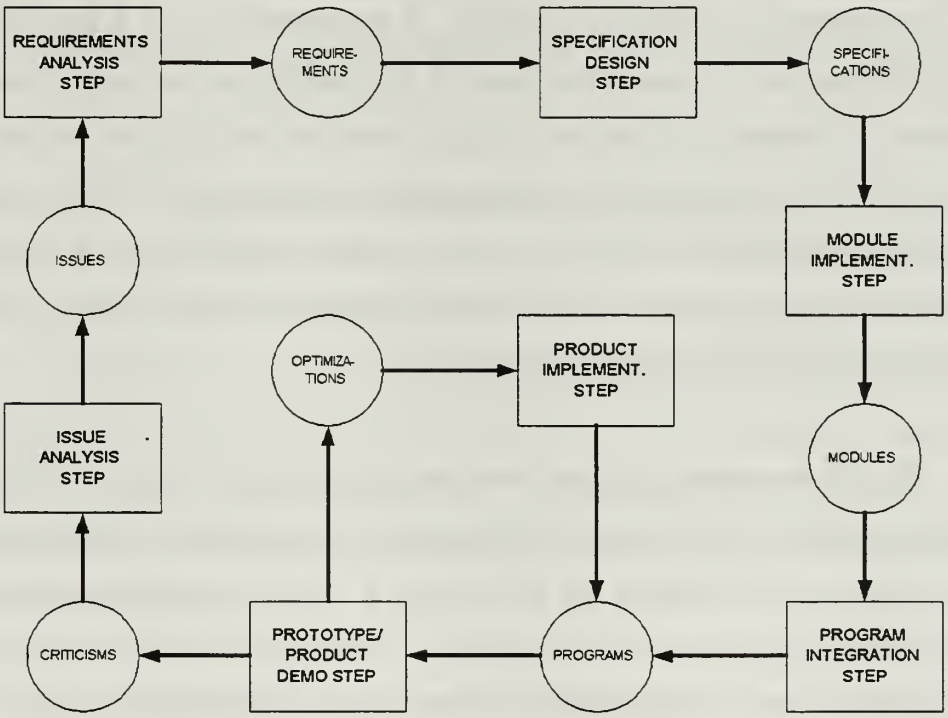


Figure 7.1: The Evolutionary Prototyping Software Process. The vertices in the graph are represented by rectangles. The arcs labeled with circles represent the edges of the digraph.

The requirements are transformed into specifications, probably in PSDL, during the specification design step. In the module implementation step, the specifications are automatically converted into code using an appropriate CASE tool, such as CAPS. The program integration step transforms the modules obtained by the generator into a

program, possibly adding code created by programmers and reusable components. This step includes integration testing and debugging. The program is demonstrated to the customer in a prototype demo step that has two possible outcomes: a) the customer is not satisfied and introduces criticisms, or b) the product matches the needs and expectations of the customer. In the first case, the process continues by analyzing the criticisms during an issue analysis step that produces new issues closing the external cycle in the graph. In the second case, the prototype contains all the required functionality, so a set of optimizations is introduced during a product implementation step. The resulting product is presented again to the customer during a product demo step closing the internal cycle of the graph.

The proposed improvement consists of the introduction of a new vertex in the graph to contain the risk assessment step. A risk assessment step can be automatically done after the completion of the specifications. From the specifications, we can derive the complexity of the product. This information is used together with personnel and organizational information and with metrics of requirements collected from the baseline to produce the risk assessment. The Relational Hypergraph Model (Harn, 1999f) enables the incorporation of this new step in the process.

The risk assessment step integrates these measures with issues in the requirements analysis steps (Fig. 7.2). The typical issues produced by the risk assessment step include the probability of developing the product by a given target time, or the estimated development time given a confidence interval. The project manager can then conduct a series of what-if analysis modifying the values of the input parameters. This information should be used at the moment of negotiate the functionality, time, and budget with the stakeholders.

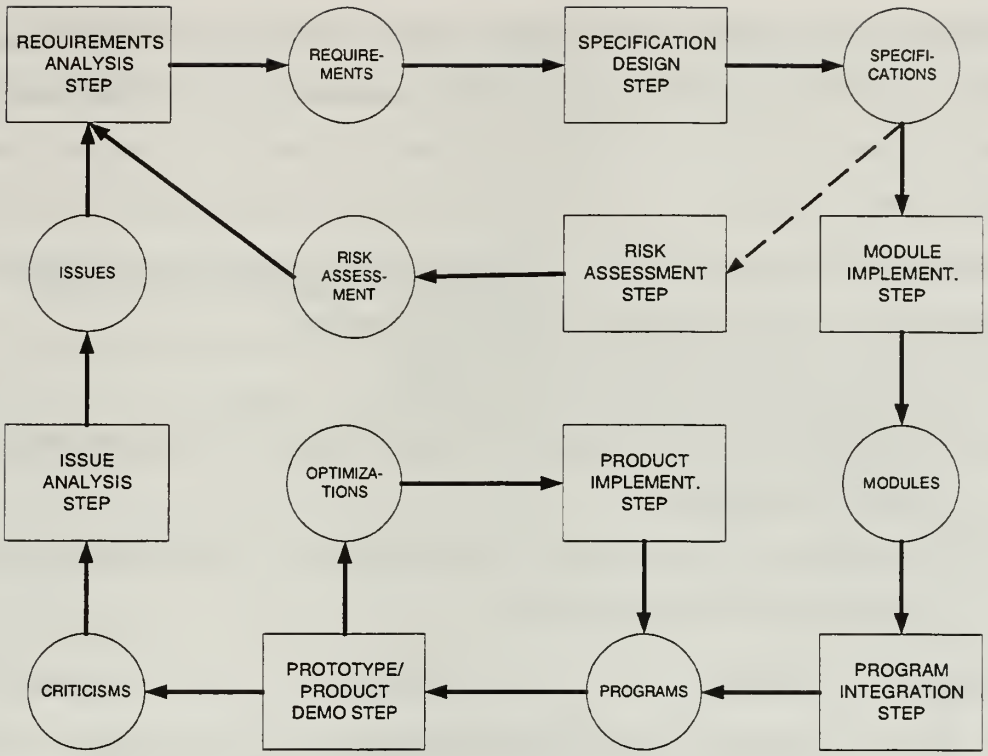


Figure 7.2: The Proposed Improvement

The development life cycle can be visualized as a succession of prototyping developments with increasing functionality followed by a final optimization that produces the system. Each of these phases has the same activity pattern, so its reasonable to model the delivery time for each one as a probability distribution from the Weibull family, but with different parameters.

During each phase a certain number of problem events occur. A problem event is an effort-consuming situation that introduces a certain amount of functional complexity to be solved (caused by a new requirement, a change in a requirement, or as the consequence of rework), and a certain amount of information exchange.

We model the occurrence of problem events in each phase as a Poisson distribution with a different mean (λ) for each phase. A random variable X has a Poisson

distribution if the probability mass function is $p(x, \lambda) = e^{-\lambda}\lambda^x / x!$ with $x = 0, 1, 2,\ldots$ for some $\lambda > 0$. (Devore, 1995). A very important application of the Poisson distribution arises in connection with the occurrence of events of a particular type over time. In this case the mean (λ) would vary over time. So, the entire development life cycle is a non-homogeneous Poisson process (Fig. 7.3). This assumption has been applied previously by (Schneidewind, 1975). We choose this distribution because:

(a) A certain rate of occurrence of events exists.

(b) The probability of more than one event occurring in a time interval depends on the length of the interval.

(c) The number of events during one time interval is independent of the number received prior to this time interval.

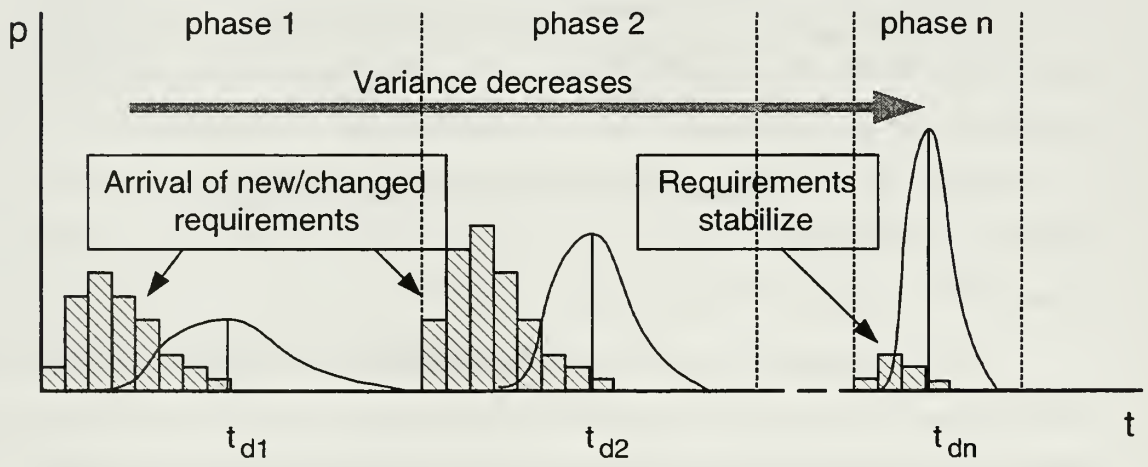


Figure 7.3: The Development Life Cycle. The shadow represents the non-homogeneous Poisson process of the problem events. The curves represent the Weibull probability distributions for the development time of each phase.

Figure 7.3 shows the different phases or evolutionary cycles in the software process. The requirements volatility tends to stabilize as a consequence of the effort in requirement elucidation. The uncertainty in the project tends to decrease because the requirement volatility decreases, the complexity to solve decreases, and the efficiency increases following learning curves. The variation on requirement volatility and efficiency affect the values of the shape parameter (α) and scale parameter (β) on the Weibull distribution. Consequently, the variance of the estimation for the duration

decreases. Figure 7.3 shows the normal evolution of the process, which is related to the normal behavior of the requirements elucidation discussed in Chapter V, Section A1 (Fig. 5.1).

B. THE RISK ASSESSMENT METHOD

The following section describes the method to assess the risk in an evolutionary software development project. The method requires the ability of collecting metrics from the project described in Chapter V. The main difference of this approach is that it is designed for software evolution. Current standards as COCOMO 81 cannot be applied in evolutionary software processes (Boehm et al, 2000). COCOMO II solves this issue but requires an estimation of the size of each evolutionary cycle, and this estimation is not easy to provide until late in each cycle. The metrics of my model can be collected since very early in the project. The objective measurement for Requirement Volatility requires one evolutionary cycle. However, Efficiency and Complexity can be objectively measured at the end of the Specification Design Step of the first evolutionary cycle. At the first cycle, the project manager can assess the risk introducing a subjective value for Requirements Volatility. For the next cycles (2, 3,... n) the metrics are totally objective.

Applying the values of complexity measured in LGC, requirements volatility, and efficiency, to the algorithms described in Chapter VI Section B, we can obtain the parameters α, β, and γ for the Weibull distribution. The estimation based on Weibull distributions can be applied since the beginning of the project with increasing accuracy as the project advances in time. At each evolutionary cycle, the decision-maker can proceed in several ways:

- Introducing a value for the time (t) in days, the decision-maker can apply Eq. 6-1 to obtain the cumulative probability for finishing the project at day t of before.
- Introducing a confidence level (p) expressed as a probability, the decision-maker can apply Eq. 6-2 to obtain the required amount of time for the project. The recommended value for p is 0.95.

- Keeping track of the requirements volatility metric, the decision-maker can detect deviations from the normal evolution of the project. This technique is explained in Chapter V, Section A.1 and Fig. 5.1.

With this information, the decision-maker can evaluate different alternatives with different values of complexity, requirements volatility, or efficiency. Usually the decision-maker cannot vary the efficiency because it depends mainly in the people, and the improvements in efficiency are not immediate. The development time is mainly sensitive to the complexity (see Chapter VI, Section A), hence negotiating the complexity for each version is the best way to increase the probability of success of the project. If the complexity cannot be reduced, then a larger development time can do the same effect. The decision-maker should plot the cumulative distribution function for each alternative and compare them using stochastic dominance as discussed in Appendix F.

The recommended estimation model is Model 4. However, if the requirements volatility is less than 68% and the application is larger than 600 LGC, the decision-maker can apply Model 3. The decision-maker should be aware that even at 95% of confidence the models could estimate short in some cases (see Fig. 6.9). However, the maximum underestimation error detected was 4%, and the maximum overestimation was not greater that 16% of the duration of the project.

Later in the development process, when all the requirements have been specified, the decision-maker can apply Eq. 6-7 to obtain a very accurate estimate of the duration of the project.

The method provides the decision-maker with formal arguments for:

- Negotiating project constraints with the stakeholders. For instance, the limitation in complexity will increase the probability of success. Alternatively, an extension in time can provide the same effect.

- Detecting issues that can make the project unfeasible. For instance, if the probability of success is very poor, and any of the constraints can be released, then it could be reasonable to abort the project before spending more money.
- Controlling the project, its resources, and its performance.

To explain the use of the method let's use a pair of examples.

Example 1. Let's suppose a software project with low efficiency, 10% of requirements volatility, and a complexity of 2500 LGC. Let's suppose also that the project should be done by day 450. By applying Model 3 we obtain the following values for the parameters of the Weibull: $\alpha = 2.5$, $\beta = 58.1$, and $\gamma = 385$. The cumulative probability for day 450, that is the probability of finishing by day 450 or before is less than 0.75. Achieving the project under this probability could be considered too risky. If the project manager can negotiate a reduction in the complexity deliverable by day 450, let's say 2000 LGC (20% of reduction), then the parameters for the Weibull change to: $\alpha = 2.5$, $\beta = 49.3$, and $\gamma = 326$. The cumulative distribution for day 450 jumps to 0.99996.

The following table shows the cumulative probabilities for different points in time. The columns titled 1.a and 1.b refer to the cumulative probabilities before and after the reduction in complexity. The same information is presented in Figure 7.4.

t	P(x<=t) 1.a	P(x<=t) 1.b
350	0.0000	0.158341
400	0.0388	0.939298
450	0.7468	0.999960
500	0.9964	1.000000
550	1.0000	1.000000

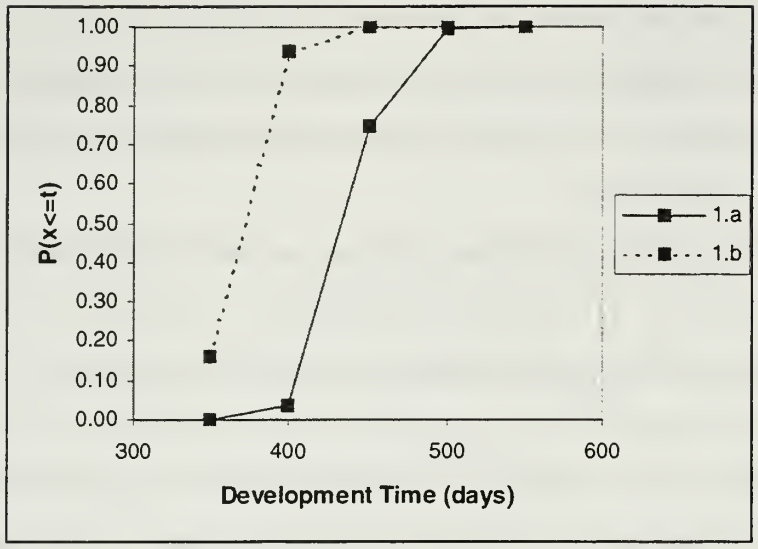


Figure 7.4: CDFs for Example 1

Example 2. Let's suppose now that the requirements volatility is 30% and the remaining characteristics do not vary. By applying Model 3 we obtain the following values for the parameters of the Weibull: $\alpha = 2.5$, $\beta = 88.7$, and $\gamma = 385$. The cumulative probability for day 450, that is the probability of finishing by day 450 or before is less than 0.4, which is really poor. A reduction in the complexity of the project from 2500 to 2000 LGC can make this project feasible. Reducing the complexity, the parameters for the Weibull change to: $\alpha = 2.5$, $\beta = 75.2$, and $\gamma = 326$. For day 450 the cumulative probability is now 0.97.

The following table shows the cumulative probabilities for different points in time. The columns titled 2.a and 2.b refer to the cumulative probabilities before and after the reduction in complexity. The same information is presented in Figure 7.5.

t	P(x<=t) 2.a	P(x<=t) 2.b
350	0.000000	0.058112
400	0.013653	0.622084
450	0.379394	0.970358
500	0.858450	0.999722
550	0.991687	1.000000

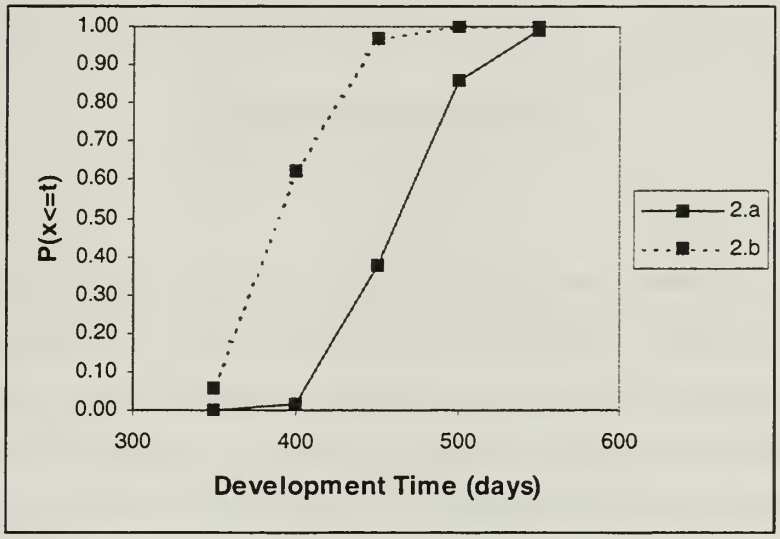


Figure 7.5: CDFs for Example 2

The method provides the stakeholders a clear visualization of the impact of their requirements, changes of requirements, and constraints, over the probability of success of the project. The model contributes to diminish the gap between the technical and non-technical stakeholders because the success of the project can be visualized in a very simple measure between zero and one.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS

This dissertation introduces a formal risk assessment model for software projects based on probabilities and metrics automatically collectable early in the project. The approach enables a project manager to evaluate the probability of success of the project very early in the life cycle. For more than twenty years the estimation standards (COCOMO 81, COCOMO II, Putnam) have been characterized by a common limitation: the requirements should be frozen in order to make estimations. This model removes this important limitation, facing the reality that requirements are inherently variable.

The problem of risk assessment for projects has been treated as unstructured. This dissertation shows a structured method to solve the problem based on metrics automatically collected from the project baselines. This contribution impacts the software engineering state of the art, as well as risk management in general. These metrics measure three risk factors identified in the research: complexity, requirements volatility, and efficiency. Each of these metrics is original and constitutes a contribution to the state of the art in software metrics. The subjectivity issue characteristic of previous research has been addressed and eliminated. Any decision-maker will arrive at the same estimates, independently of his or her expertise.

The model is perfectly suited for any evolutionary software process because it follows the same philosophy. The risk assessment and estimations are conducted at each evolutionary cycle with increasing knowledge and decreasing variance. The research formalizes an improvement in the evolutionary software process, introducing a risk assessment step that can be automated.

The dissertation suggests that software processes can be improved using more flexible organizational designs. This observation based on the research of Perrow, Burton and Obel (Burton & Obel, 1998) contradicts the position of the SEI. This controversial issue requires future study and remains as a future line of research. The dissertation also shows that the current standard planning techniques, such as Pert, Gantt, and CPM, could result in overly optimistic results when they are applied to communication-intensive projects like software development.

Finally, the research is based on simulations and a small set of real projects. It is desirable to collect and analyze metrics and completion times of a larger set of real software projects to confirm and refine the models.

APPENDIX A
ANALYSIS WITH ORGANIZATIONAL CONSULTANT

The following reports were produced using Organizational Consultant expert system. The first report analyzes a fictive organization "Software Engineering" which represents a typical public software development department. The appendix has four reports analyzing different CMM levels (2, 3, 1, and 5). The findings for levels 2 and 3 were used in the configuration of the simulations. The findings for levels 1 and 5 were used to test the impact of the organizational parameters of VitéProject.

The results of this appendix were used to calibrate the organizational design parameters of VitéProject. These parameters are: Centralization, Formalization, and Matrix Strength (see Fig. 4.1 in Chapter IV). The validation process is discussed in Chapter IV Section D.

REPORT SUMMARY - Software Engineering (CMM 2)

Time: 2:17:22 PM, 12/29/99
Scenario: Scenario 1

INPUT DATA SUMMARY

The description below summarizes and interprets your answers to the questions about your organization and its situation. It states your answers concerning the organization's current configuration, complexity, formalization, and centralization. Your responses to the various questions on the contingencies of age, size, technology, environment, management style, cultural climate and strategy factors are also given. The writeup below summarizes the input data for the analysis.

- Software Engineering has an adhocracy configuration (cf 100).
- Software Engineering has a small number of different jobs (cf 100).
- Of the employees at Software Engineering 76 to 100 % have an advanced degree or many years of special training (cf 100).
- Software Engineering has 3 to 5 vertical levels separating top management from the bottom level of the organization (cf 100).
- The mean number of vertical levels is 3 to 5 (cf 100).
- Software Engineering has 1 or 2 separate geographic locations (cf 100).
- Software Engineering's average distance of these separate units from the organization's headquarters is of no relevance because there is only one site undetermined (cf 100).
- An undetermined number of Software Engineering's total workforce is located at these separate units (cf 100).
- Job descriptions are available for none or an undetermined number of employees (cf 100).
- Where written job descriptions exist, the employees are supervised an undetermined manner to ensure compliance with standards set in the job description (cf 100).
- The employees are allowed to deviate in an undetermined way from the standards (cf 100).
- 0 to 20 % non-managerial employees are given written operating instructions or procedures for their job (cf 100).
- The written instructions or procedures given are of no relevance as there are no written instructions or they may be undetermined (cf 100).
- Supervisors and middle managers are to some extent free from rules, procedures, and policies when they make decisions (cf 100).
- Less than 20 % of all the rules and procedures that exist within the organization are in writing (cf 100).

- Top Management is to a great extent involved in gathering the information they will use in making decisions (cf 100).
- Top management participates in the interpretation of 61 to 80 % of the information input (cf 100).
- Top management directly controls 21 to 40 % of the decisions executed (cf 100).
- The typical middle manager has little discretion over establishing his or her budget (cf 100).
- The typical middle manager has little discretion over how his/her unit will be evaluated (cf 100).
- The typical middle manager has little discretion over the hiring and firing of personnel (cf 100).
- The typical middle manager has little discretion over personnel rewards - (ie, salary increases and promotions) (cf 100).
- The typical middle manager has some discretion over purchasing equipment and supplies (cf 100).
- The typical middle manager has little discretion over establishing a new project or program (cf 100).
- The typical middle manager has very great discretion over how work exceptions are to be handled (cf 100).
- Software Engineering has 25 employees (cf 100).
- Software Engineering's age is young (cf 100).
- Software Engineering's ownership status is public (cf 100).
- Software Engineering has some different products (cf 100).
- Software Engineering has few different markets (cf 100).
- Software Engineering only operates in one country (cf 100).
- Software Engineering has no different products in the foreign markets (cf 100).
- Software Engineering's major activity is categorized as service (cf 100).
- Software Engineering has a specialized customer-oriented service technology (cf 100).
- Software Engineering has undetermined technology (cf 100).
- Software Engineering's technology is undetermined with respect to divisibility (cf 100).
- Software Engineering's technology dominance is strong (cf 100).
- Software Engineering has given no information about a possible advanced information system (cf 100).
- Software Engineering's environment is complex (cf 100).
- The uncertainty of Software Engineering's environment is high (cf 100).
- The equivocality of the organization's environment is high (cf 100).
- Software Engineering's environment has an undetermined level of hostility (cf 100).
- Top management prefers to make resource allocations and detailed operating decisions (cf 100).
- Top management primarily prefers to make long-term decisions (cf 100).
- Top management has a preference for very aggregate information when making decisions (cf 100).
- Top management has a preference for some proactive actions and some reactive actions (cf 100).
- Top management is risk averse (cf 100).
- Top management has a preference for a combination of motivation and control (cf 100).
- Software Engineering operates in an industry with a medium capital requirement (cf 100).
- Software Engineering has a high product innovation (cf 100).
- Software Engineering has a high process innovation (cf 100).
- Software Engineering has a high concern for quality (cf 100).
- Software Engineering's price level is undetermined relative to its competitors (cf 100).
- The level of trust is high (cf 100).
- The level of conflict is low (cf 100).
- The employee morale is not known (cf 100).
- Rewards are given in a not known fashion (cf 100).
- The resistance to change is not known (cf 100).
- The leader credibility is high (cf 100).
- The level of scapegoating is low (cf 100).

THE SIZE

The size of the organization - large, medium, or small - is based upon the number of employees, adjusted for their level of education or technical skills.
Based on the answers you provided, it is most likely that your organization's size is medium (cf 50).
More than 75 % of the people employed by Software Engineering have a high level of education. Adjustments are made to this effect. The adjusted number of employees is lower than 500 but greater than

100 and Software Engineering is categorized as medium. However, for this adjusted number this size does not have a major effect on the organizational structure.

THE CLIMATE

The organizational climate effect is the summary measure of people and behavior.
Based on the answers you provided, it is most likely that the organizational climate is a group climate (cf 76).
It could also be that climate is a developmental (cf 73).
The group climate is characterized as a friendly place to work where people share a lot of themselves. It is like an extended family. The leaders, or head of the organization, are considered to be mentors and, perhaps even parent figures. The organization is held together by loyalty or tradition. Commitment is high. The organization emphasizes the long-term benefit of human resource development with high cohesion and morale being important. Success is defined in terms of sensitivity to customers and concern for people. The organization places a premium on teamwork, participation, and consensus.
When the organization has a high level of trust it is likely that the organization has a group climate. An organization with little conflict can be categorized to have group climate. High leader credibility characterizes an organization with a group climate. An organization with a low level of scapegoating may have a group climate.
The developmental climate is characterized as a dynamic, entrepreneurial and creative place to work. People stick their necks out and take risks. The leaders are considered to be innovators and risk takers. The glue that holds organizations together is commitment to experimentation and innovation. The emphasis is on being on the leading edge. Readiness for change and meeting new challenges are important. The organization's long-term emphasis is on growth and acquiring new resources. Success means having unique and new products or services and being a product or service leader is important. The organization encourages individual initiative and freedom.
When the organization has a high to medium level of trust it is likely that the organization has a developmental climate. An organization with low level of conflict can be categorized to have a developmental climate. Medium to high leader credibility characterizes an organization with a developmental climate. An organization with a medium level of scapegoating may have a developmental climate.

THE MANAGEMENT STYLE

The level of management's microinvolvement in decision making is the summary measure of management style. Leaders have a low preference for microinvolvement; managers have a high preference for microinvolvement.
Based on the answers you provided, it is most likely that your management profile has a medium preference for microinvolvement (cf 78).
The management of Software Engineering has a preference for letting some decisions be made by other managers. This will lead toward a medium preference for microinvolvement. The management of Software Engineering has a preference for taking actions on some decisions and being reactive toward others. This will lead toward a medium preference for microinvolvement. Management has a preference for using both motivation and control to coordinate the activities, which leads toward a medium preference for microinvolvement.

THE STRATEGY

The organization's strategy is categorized as one of either prospector, analyzer with innovation, analyzer without innovation, defender, or reactor. These categories follow Miles and Snow's typology. Based on your answers, the organization has been assigned to a strategy category. This is a statement of the current strategy; it is not an analysis of what is the best or preferred strategy for the organization.
Based on the answers you provided, it is most likely that your organization's strategy is an analyzer with innovation strategy (cf 68).
It could also be: a prospector (cf 64).

An organization with an analyzer with innovation strategy is an organization that combines the strategy of the defender and the prospector. It moves into the production of a new product or enters a new market after viability has been shown. But in contrast to an analyzer without innovation, it has innovations that run concurrently with the regular production. It has a dual technology core.
An organization with a medium capital investment is likely to have some capabilities rather fixed, but can also adjust. The analyzer with innovation which seeks new opportunities but also maintains its profitable position is appropriate. With a concern for high quality an analyzer with innovation strategy is a likely strategy for Software Engineering. With top management preferring a medium level of microinvolvement top management wants some influence. This can be obtained via control over current operations. Product innovation should be less controlled. The strategy is therefore likely to be analyzer with innovation.
An organization with a prospector strategy is an organization that continually searches for market opportunities and regularly experiments with potential responses to emerging environmental trends. Thus, the organization is often the creator of change and uncertainty to which its competitors must respond. However, because of its strong concern for product and market innovation, a prospector usually is not completely efficient.
With a concern for high quality a prospector strategy is a likely strategy for Software Engineering.

THE CURRENT ORGANIZATIONAL CHARACTERISTICS

Based on your answers, the organization's complexity, formalization, and centralization have been calculated. This is the current organization. Later in this report, there will be recommendations for the organization.
The current organizational complexity is medium (cf 100).
The current horizontal differentiation is medium (cf 100).
The current vertical differentiation is low (cf 100).
The current spatial differentiation is low (cf 100).
The current centralization is medium (cf 100).
The current formalization is low (cf 100).
The current organization has been categorized with respect to formalization, centralization, and complexity.
The categorization is based on the input you gave and does not take missing information into account.

SITUATION MISFITS

A situation misfit is an unbalanced situation among the contingency factors of management style, size, environment, technology, climate, and strategy.
The following misfits are present: (cf 100).
Software Engineering has both an analyzer strategy and few products. Generally, more products are required for an analyzer. A few products may be reasonable in the short run, but an analyzer should be in constant consideration of new possibilities. When a few, unchanging products become the norm, the analyzer should broaden its scope of new opportunities.

ORGANIZATIONAL CONSULTANT RECOMMENDATIONS

Based on your answers about the organization, its situation, and the conclusions with the greatest certainty factor from the analyses above Organizational Consultant has derived recommendations for the organization's configuration, complexity, formalization, and centralization. There are also recommendations for coordination and control, the appropriate media richness for communications, and incentives. More detailed recommendations for possible changes in the current organization are also provided.

ORGANIZATIONAL CONFIGURATIONS

The most likely configuration that best fits the situation has been estimated to be a matrix configuration (cf 59).

A matrix structure is a structure that assigns specialists from functional departments to work on one or more interdisciplinary teams that are led by project leaders. Permanent product teams are also possible. A dual hierarchy manages the same activities and individuals at the same time.
When Software Engineering's environment has neither low equivocality nor low complexity, the configuration should be matrix. When Software Engineering is of medium size, the configuration can be a matrix configuration. The matrix configuration is a more likely configuration when Software Engineering has a unit production technology.

ORGANIZATIONAL CHARACTERISTICS

The recommended degree of organizational complexity is medium (cf 43).
Medium size organizations should have medium organizational complexity. Top management of Software Engineering has a preference for a medium level of microinvolvement, which drives the organizational complexity towards medium. A group climate in the organization requires a medium level of complexity with a low level of vertical differentiation.
The recommended degree of horizontal differentiation is low (cf 28).
It, too, could be: medium (cf 19).
The recommended degree of vertical differentiation is low (cf 38).
The recommended degree of formalization is medium (cf 48).
There should be some formalization between the organizational units but less formalization within the units due to the high professionalization. Software Engineering has a medium capital requirement, which leads to medium formalization. Medium size organizations should have medium formalization. Medium formalization is consistent with the leadership style when top management's preference for microinvolvement is neither very great nor very low.
The recommended degree of centralization is medium (cf 45).
Software Engineering has an analyzer with innovation strategy. Centralization should be medium. There should be tight control over current activities and looser control over new ventures. Software Engineering is of medium size. Such organizations should have medium to high centralization. Medium centralization is recommended when top management has neither a great desire nor very little desire for microinvolvement.
Software Engineering's span of control should be narrow (cf 30).
It, too, at places should be moderate (cf 25).
Since Software Engineering has a nonroutine technology, it should have a narrow span of control.
Software Engineering should use media with high media richness (cf 85).
The information media that Software Engineering uses should provide a large amount of information (cf 85).
Incentives should be based on results (cf 85).
Software Engineering should use an undetermined process as means for coordination and control (cf 100).
When the environment of Software Engineering has high equivocality, high uncertainty, and high complexity, coordination and control should be obtained through integrators and group meetings. The richness of the media should be high with a large amount of information. Incentives must be results based. Coordination is a major issue requiring a lot of time by functional managers and product or project managers. Managers should make frequent adjustments in order to maintain project and product goals and use scarce functional resources and personnel efficiently. In an international firm, matrix dimensions will likely include country or region and may include either product, customer, or function. Project or product managers will likely be required to champion new innovations in customers, products or technologies.
When the organization has a group climate, coordination should be obtained using integrators and group meetings. Incentives could be results based but with a group orientation. An organization with a group climate will likely have to process a large amount of information and will need information media with high richness.

ORGANIZATIONAL MISFITS

Organizational misfits compares the recommended organization with the current organization.
The following organizational misfits are present: (cf 100).

Current and prescribed configuration do not match.
Current and prescribed formalization do not match.

MORE DETAILED RECOMMENDATIONS

There are a number of more detailed recommendations (cf 100).
You may consider increasing the number of positions for which job descriptions are available.
You may consider supervising the employees more closely.
You may consider allowing employees less latitude from standards.
You may consider more written job descriptions.
Managerial employees may be asked to follow written instructions and procedures more closely.
You may consider having more written rules and procedures.

END

REPORT SUMMARY - Software Engineering (CMM 3)

Time: 2:40:37 PM, 12/29/99
Scenario: Scenario 2

INPUT DATA SUMMARY

The description below summarizes and interprets your answers to the questions about your organization and its situation. It states your answers concerning the organization's current configuration, complexity, formalization, and centralization. Your responses to the various questions on the contingencies of age, size, technology, environment, management style, cultural climate and strategy factors are also given. The writeup below summarizes the input data for the analysis.
- Software Engineering has a matrix configuration (cf 100).
- Software Engineering has a small number of different jobs (cf 100).
- Of the employees at Software Engineering 76 to 100 % have an advanced degree or many years of special training (cf 100).
- Software Engineering has 3 to 5 vertical levels separating top management from the bottom level of the organization (cf 100).
- The mean number of vertical levels is 3 to 5 (cf 100).
- Software Engineering has 1 or 2 separate geographic locations (cf 100).
- Software Engineering's average distance of these separate units from the organization's headquarters is of no relevance because there is only one site undetermined (cf 100).
- An undetermined number of Software Engineering's total workforce is located at these separate units (cf 100).
- Job descriptions are available for operational employees, low and middle management (cf 100).
- Where written job descriptions exist, the employees are supervised closely to ensure compliance with standards set in the job description (cf 100).
- The employees are allowed to deviate a moderate amount from the standards (cf 100).
- 81 to 100 % non-managerial employees are given written operating instructions or procedures for their job (cf 100).
- The written instructions or procedures given are followed to a great extent (cf 100).
- Supervisors and middle managers are to a little extent free from rules, procedures, and policies when they make decisions (cf 100).
- More than 80 % of all the rules and procedures that exist within the organization are in writing (cf 100).
- Top Management is to some extent involved in gathering the information they will use in making decisions (cf 100).
- Top management participates in the interpretation of 41 to 60 % of the information input (cf 100).
- Top management directly controls 0 to 20 % of the decisions executed (cf 100).

- The typical middle manager has some discretion over establishing his or her budget (cf 100).
- The typical middle manager has some discretion over how his/her unit will be evaluated (cf 100).
- The typical middle manager has great discretion over the hiring and firing of personnel (cf 100).
- The typical middle manager has some discretion over personnel rewards - (ie, salary increases and promotions) (cf 100).
- The typical middle manager has some discretion over purchasing equipment and supplies (cf 100).
- The typical middle manager has some discretion over establishing a new project or program (cf 100).
- The typical middle manager has very great discretion over how work exceptions are to be handled (cf 100).
- Software Engineering has 25 employees (cf 100).
- Software Engineering's age is young (cf 100).
- Software Engineering's ownership status is public (cf 100).
- Software Engineering has few different products (cf 100).
- Software Engineering has few different markets (cf 100).
- Software Engineering only operates in one country (cf 100).
- Software Engineering has no different products in the foreign markets (cf 100).
- Software Engineering's major activity is categorized as service (cf 100).
- Software Engineering has a specialized customer-oriented service technology (cf 100).
- Software Engineering has a medium routine technology (cf 100).
- Software Engineering's technology is highly divisible (cf 100).
- Software Engineering's technology dominance is strong (cf 100).
- Software Engineering has either planned or already has an advanced information system (cf 100).
- Software Engineering's environment is complex (cf 100).
- The uncertainty of Software Engineering's environment is high (cf 100).
- The equivocality of the organization's environment is high (cf 100).
- Software Engineering's environment has an undetermined level of hostility (cf 100).
- Top management prefers to make policy and general resource allocation decisions (cf 100).
- Top management primarily prefers to make long-term decisions (cf 100).
- Top management has a preference for very aggregate information when making decisions (cf 100).
- Top management has a preference for some proactive actions and some reactive actions (cf 100).
- Top management is risk averse (cf 100).
- Top management has a preference for high control (cf 100).
- Software Engineering operates in an industry with a medium capital requirement (cf 100).
- Software Engineering has a high product innovation (cf 100).
- Software Engineering has a high process innovation (cf 100).
- Software Engineering has a high concern for quality (cf 100).
- Software Engineering's price level is undetermined relative to its competitors (cf 100).
- The level of trust is high (cf 100).
- The level of conflict is low (cf 100).
- The employee morale is high (cf 100).
- Rewards are given in a inequitably fashion (cf 100).
- The resistance to change is not known (cf 100).
- The leader credibility is high (cf 100).
- The level of scapegoating is low (cf 100).

THE SIZE

The size of the organization - large, medium, or small - is based upon the number of employees, adjusted for their level of education or technical skills.
Based on the answers you provided, it is most likely that your organization's size is medium (cf 50).
More than 75 % of the people employed by Software Engineering have a high level of education. Adjustments are made to this effect. The adjusted number of employees is lower than 500 but greater than 100 and Software Engineering is categorized as medium. However, for this adjusted number this size does not have a major effect on the organizational structure.

THE CLIMATE

The organizational climate effect is the summary measure of people and behavior.
Based on the answers you provided, it is most likely that the organizational climate is a group climate (cf 82).
It could also be the that climate is a developmental (cf 80).

The group climate is characterized as a friendly place to work where people share a lot of themselves. It is like an extended family. The leaders, or head of the organization, are considered to be mentors and, perhaps even parent figures. The organization is held together by loyalty or tradition. Commitment is high. The organization emphasizes the long-term benefit of human resource development with high cohesion and morale being important. Success is defined in terms of sensitivity to customers and concern for people. The organization places a premium on teamwork, participation, and consensus.
When the organization has a high level of trust it is likely that the organization has a group climate. An organization with little conflict can be categorized to have group climate. Employees with a high morale is one element of group climate. High leader credibility characterizes an organization with a group climate. An organization with a low level of scapegoating may have a group climate.
The developmental climate is characterized as a dynamic, entrepreneurial and creative place to work. People stick their necks out and take risks. The leaders are considered to be innovators and risk takers. The glue that holds organizations together is commitment to experimentation and innovation. The emphasis is on being on the leading edge. Readiness for change and meeting new challenges are important. The organization's long-term emphasis is on growth and acquiring new resources. Success means having unique and new products or services and being a product or service leader is important. The organization encourages individual initiative and freedom.
When the organization has a high to medium level of trust it is likely that the organization has a developmental climate. An organization with low level of conflict can be categorized to have a developmental climate. Employees with a high morale is frequently one element of a developmental climate. Medium to high leader credibility characterizes an organization with a developmental climate. An organization with a medium level of scapegoating may have a developmental climate.

THE MANAGEMENT STYLE

The level of management's microinvolvement in decision making is the summary measure of management style. Leaders have a low preference for microinvolvement; managers have a high preference for microinvolvement.
Based on the answers you provided, it is most likely that your management profile has a low preference for microinvolvement (cf 72).
It could also be that your management profile has an inappropriate preference (cf 70).
It could also be that your management profile has a high preference (cf 69).
The management of Software Engineering has a preference for delegating decisions. This will lead toward a low preference for microinvolvement. Management has a long-term horizon when making decisions, which characterizes a preference for a low microinvolvement. Since the management has a preference for making decisions on the basis of very aggregate information a low preference for microinvolvement characterization is appropriate.
The management dimensions are not in balance. This is likely to result in an ineffectual individual.
Management is risk averse. This is one of the characteristics of a manager with a high preference for microinvolvement. Management has a preference for using control to coordinate activities, which leads toward a high preference for microinvolvement.

THE STRATEGY

The organization's strategy is categorized as one of either prospector, analyzer with innovation, analyzer without innovation, defender, or reactor. These categories follow Miles and Snow's typology. Based on your answers, the organization has been assigned to a strategy category. This is a statement of the current strategy; it is not an analysis of what is the best or preferred strategy for the organization.

Based on the answers you provided, it is most likely that your organization's strategy is a prospector strategy (cf 73).
It could also be: a defender (cf 72).
It could also be: an analyzer with innovation (cf 72).
An organization with a prospector strategy is an organization that continually searches for market opportunities and regularly experiments with potential responses to emerging environmental trends. Thus, the organization is often the creator of change and uncertainty to which its competitors must respond. However, because of its strong concern for product and market innovation, a prospector usually is not completely efficient.
With a concern for high quality a prospector strategy is a likely strategy for Software Engineering. With top management preferring a relatively low level of microinvolvement, the strategy is likely to be prospector.
An organization with a defender strategy is an organization that has a narrow product market domain. Top managers in this type of organization are expert in their organization's limited area of operation but do not tend to search outside their domains for new opportunities. As a result of this narrow focus, these organizations seldom need to make major adjustments in their technology, structure, or methods of operation. Instead, they devote primary attention to improving the efficiency of their existing operations.
Software Engineering has few products. It needs to defend these products well in the marketplace. Viability depends on being successful with these limited activities. With a concern for high quality a defender strategy is a likely strategy for Software Engineering.
An organization with an analyzer with innovation strategy is an organization that combines the strategy of the defender and the prospector. It moves into the production of a new product or enters a new market after viability has been shown. But in contrast to an analyzer without innovation, it has innovations that run concurrently with the regular production. It has a dual technology core.
An organization with a medium capital investment is likely to have some capabilities rather fixed, but can also adjust. The analyzer with innovation which seeks new opportunities but also maintains its profitable position is appropriate. For a medium routine technology, Software Engineering has some flexibility. It is consistent with an analyzer with innovation strategy. With a concern for high quality an analyzer with innovation strategy is a likely strategy for Software Engineering.

THE CURRENT ORGANIZATIONAL CHARACTERISTICS

Based on your answers, the organization's complexity, formalization, and centralization have been calculated. This is the current organization. Later in this report, there will be recommendations for the organization.
The current organizational complexity is medium (cf 100).
The current horizontal differentiation is medium (cf 100).
The current vertical differentiation is low (cf 100).
The current spatial differentiation is low (cf 100).
The current centralization is medium (cf 100).
The current formalization is high (cf 100).
The current organization has been categorized with respect to formalization, centralization, and complexity. The categorization is based on the input you gave and does not take missing information into account.

SITUATION MISFITS

A situation misfit is an unbalanced situation among the contingency factors of management style, size, environment, technology, climate, and strategy.
The following misfits are present: (cf 100).
Software Engineering has both a prospector strategy and a risk adverse management. This strategy conflicts with the management's risk adverse attitude. A prospector strategy demands a projection into the unknown with new and innovative products and services, where the returns are uncertain. A risk adverse management will be very uncomfortable with this high level of risk. Risk adverse managers prefer situations with less uncertainty. It is possible to either change the prospector strategy or hire more risk assuming managers. Usually a risk adverse management will control expenditures to reduce or eliminate

the prospector projects. If the environment and markets call for a prospector strategy, a new management would be preferable. Some risk adverse managers can adapt, but it is very difficult.
Software Engineering has both a prospector strategy and not many products or markets. The prospector will create a broad range of new possible products and services, which requires a large number of possible products and markets. A prospector requires variety to explore and find new products and markets for its innovations. With limited product and market opportunity, the range of prospector possibilities may exceed the environmental possibilities. The prospector needs to seek new markets as well as new products. If the markets do not exist or cannot be created, the prospector will incur high costs of innovation without return.
Software Engineering has a group climate. This is a mismatch with a prospector strategy! A group climate has low resistance to change. A prospector strategy is committed to changes.

ORGANIZATIONAL CONSULTANT RECOMMENDATIONS

Based on your answers about the organization, its situation, and the conclusions with the greatest certainty factor from the analyses above Organizational Consultant has derived recommendations for the organization's configuration, complexity, formalization, and centralization. There are also recommendations for coordination and control, the appropriate media richness for communications, and incentives. More detailed recommendations for possible changes in the current organization are also provided.

ORGANIZATIONAL CONFIGURATIONS

The most likely configuration that best fits the situation has been estimated to be an adhocracy configuration (cf 68).
It is certainly not: a professional bureaucracy (cf -73).
It is certainly not: a machine bureaucracy (cf -73).
An adhocracy organization is normally an organization with high horizontal differentiation, low vertical differentiation, low formalization, decentralization, and great flexibility and responsiveness.
An adhocracy configuration is appropriate when neither the environmental equivocality of Software Engineering nor the environmental uncertainty is low. When the organization is also young, the conclusion that it should bean adhocracy is further strengthened. Since top management has a low preference for microinvolvement, the ad hoc configuration is feasible. However, the size of the organization is not very important for the choice of an adhocracy configuration. A prospector like Software Engineering should be configured as an ad hoc organization. An organization with a group climate could have an ad hoc configuration.
Since the organization has a prospector strategy, it cannot have a configuration like a professional bureaucracy.
When the organization has a prospector strategy, it cannot be a machine bureaucracy!

ORGANIZATIONAL CHARACTERISTICS

The recommended degree of organizational complexity is medium (cf 54).
Medium size organizations should have medium organizational complexity. Software Engineering has a technology that is somewhat routine, which implies that the organizational complexity should be medium. Because Software Engineering has an advanced information system, organizational complexity can be greater than it could otherwise. A group climate in the organization requires a medium level of complexity with a low level of vertical differentiation.
The recommended degree of horizontal differentiation is low (cf 34).
It, too, could be: medium (cf 24).
The recommended degree of vertical differentiation is low (cf 72).
It, too, could be: medium (cf 62).
The recommended degree of formalization is low (cf 56).
Software Engineering has a prospector strategy. A low formalization is required so that the organization can react quickly. Low formalization is also required because of the need for innovations. Since the set of variables in the environment that will be important is not known and since it is not possible to predict what will happen, no efficient rules and procedures can be developed, which implies that Software Engineering's

formalization should be low. Low formalization is consistent with top management having a low preference for microinvolvement. A group climate in the organization requires a low level of formalization.
The recommended degree of centralization is low (cf 46).
There is evidence against it should be: high (cf -16).
Software Engineering has a prospector strategy. A low centralization is required so that the organization can react and innovate quickly. Since there are many factors in the environment that affect the organization but Software Engineering does not know which factors are or will be important for Software Engineering, centralization should be low. Low centralization can be allowed when top management has no desire for microinvolvement. A group climate in the organization requires a low level of centralization.
Software Engineering's span of control should be moderate (cf 62).
Since Software Engineering has some technology routineness, it should have a moderate span of control.
Software Engineering should use media with high media richness (cf 85).
The information media that Software Engineering uses should provide a large amount of information (cf 85).
Incentives should be based on results (cf 85).
Software Engineering should use meetings as means for coordination and control (cf 94).
When the environment of Software Engineering has high equivocality, high uncertainty, and high complexity, coordination and control should be obtained through integrators and group meetings. The richness of the media should be high with a large amount of information. Incentives must be results based. An open organizational climate and team spirit must be fostered. Information must be shared among all levels. Constructive conflict on 'what to do' will be usual. Individual tolerance of ambiguity and uncertainty will be necessary. Individual performance evaluation will be problematic and largely subjective. Mutual adjustments of 'give and take' will be the norm. Frequent informal meetings and temporary task forces will be the primary coordinating devices. When the organization has a group climate, coordination should be obtained using integrators and group meetings. Incentives could be results based but with a group orientation. An organization with a group climate will likely have to process a large amount of information and will need information media with high richness.

ORGANIZATIONAL MISFITS

Organizational misfits compare the recommended organization with the current organization.
The following organizational misfits are present: (cf 100).
Current and prescribed configuration do not match.
Current and prescribed centralization do not match.
Current and prescribed formalization do not match.

MORE DETAILED RECOMMENDATIONS

There are a number of more detailed recommendations (cf 100).
You may consider supervising the employees less closely.
You may consider fewer written job descriptions.
Managerial employees may be asked to pay less attention to written instructions and procedures.
You may give supervisors and middle managers fewer rules and procedures.
You may consider having fewer rules and procedures put in writing.

END

REPORT SUMMARY - CMM Level 1

Time: 4:46:58 PM, 9/6/2000
Scenario: Scenario1

INPUT DATA SUMMARY

The description below summarizes and interprets your answers to the questions about your organization and its situation. It states your answers concerning the organization's current configuration, complexity, formalization, and centralization. Your responses to the various questions on the contingencies of age, size, technology, environment, management style, cultural climate and strategy factors are also given. The writeup below summarizes the input data for the analysis.

- CMM Level 1 has a simple configuration (cf 100).
- CMM Level 1 has an undetermined number of different jobs (cf 100).
- Of the employees at CMM Level 1 0 to 10 % have an advanced degree or many years of special training (cf 100).
- CMM Level 1 has 1 or 2 vertical levels separating top management from the bottom level of the organization (cf 100).
- The mean number of vertical levels is 1 or 2 (cf 100).
- CMM Level 1 has 1 or 2 separate geographic locations (cf 100).
- CMM Level 1's average distance of these separate units from the organization's headquarters is of no relevance because there is only one site undetermined (cf 100).
- An undetermined number of CMM Level 1's total workforce is located at these separate units (cf 100).
- Job descriptions are available for none or an undetermined number of employees (cf 100).
- Where written job descriptions exist, the employees are supervised very loosely to ensure compliance with standards set in the job description (cf 100).
- The employees are allowed to deviate a great deal from the standards (cf 100).
- 0 to 20 % non-managerial employees are given written operating instructions or procedures for their job (cf 100).
- The written instructions or procedures given are followed a little (cf 100).
- Supervisors and middle managers are to a very great extent free from rules, procedures, and policies when they make decisions (cf 100).
- Less than 20 % of all the rules and procedures that exist within the organization are in writing (cf 100).
- Top Management is only a little involved in gathering the information they will use in making decisions (cf 100).
- Top management participates in the interpretation of an undetermined percentage of the information input (cf 100).
- Top management directly controls more than 80 % of the decisions executed (cf 100).
- The typical middle manager has no discretion over establishing his or her budget (cf 100).
- The typical middle manager has no discretion over how his/her unit will be evaluated (cf 100).
- The typical middle manager has no discretion over the hiring and firing of personnel (cf 100).
- The typical middle manager has no discretion over personnel rewards - (ie, salary increases and promotions) (cf 100).
- The typical middle manager has little discretion over purchasing equipment and supplies (cf 100).
- The typical middle manager has no discretion over establishing a new project or program (cf 100).
- The typical middle manager has great discretion over how work exceptions are to be handled (cf 100).
- CMM Level 1 has 10 employees (cf 100).
- CMM Level 1's age is young (cf 100).
- CMM Level 1's ownership status is undetermined (cf 100).
- CMM Level 1 has few different products (cf 100).
- CMM Level 1 has few different markets (cf 100).
- CMM Level 1 only operates in one country (cf 100).
- CMM Level 1 has no different products in the foreign markets (cf 100).
- CMM Level 1's major activity is categorized as service (cf 100).
- CMM Level 1 has a specialized customer-oriented service technology (cf 100).

- CMM Level 1 has a nonroutine technology (cf 100).
- CMM Level 1's technology is undetermined with respect to divisibility (cf 100).
- CMM Level 1's technology dominance is strong (cf 100).
- CMM Level 1 has no advanced information system (cf 100).
- CMM Level 1's environment is complex (cf 100).
- The uncertainty of CMM Level 1's environment is high (cf 100).
- The equivocality of the organization's environment is high (cf 100).
- CMM Level 1's environment has a medium hostility (cf 100).
- Top management prefers to make general decisions as well as detailed operating decisions (cf 100).
- Top management primarily prefers to make short-time decisions (cf 100).
- Top management has a preference for very detailed information when making decisions (cf 100).
- Top management has a preference for reactive actions (cf 100).
- Top management is risk neutral (cf 100).
- Top management has a preference for motivation and control that is unknown (cf 100).
- CMM Level 1 operates in an industry with a medium capital requirement (cf 100).
- CMM Level 1 has a low product innovation (cf 100).
- CMM Level 1 has a low process innovation (cf 100).
- CMM Level 1 has a low concern for quality (cf 100).
- CMM Level 1's price level is undetermined relative to its competitors (cf 100).
- The level of trust is not known (cf 100).
- The level of conflict is not known (cf 100).
- The employee morale is not known (cf 100).
- Rewards are given in a not known fashion (cf 100).
- The resistance to change is not known (cf 100).
- The leader credibility is not known (cf 100).
- The level of scapegoating is not known (cf 100).

THE SIZE

The size of the organization - large, medium, or small - is based upon the number of employees, adjusted for their level of education or technical skills.

Based on the answers you provided, it is most likely that your organization's size is small (cf 100).

Less than 20% of the people employed by CMM Level 1 have a high level of education. No adjustments for educational level are made. The adjusted number of employees is lower than 100 and CMM Level 1 is categorized as small.

THE CLIMATE

The organizational climate effect is the summary measure of people and behavior.
Based on the answers you provided, it is most likely that the organizational climate is a unknown climate (cf 100).

THE MANAGEMENT STYLE

The level of management's microinvolvement in decision making is the summary measure of management style. Leaders have a low preference for microinvolvement; managers have a high preference for microinvolvement.

Based on the answers you provided, it is most likely that your management profile has a high preference for microinvolvement (cf 92).

The management of CMM Level 1 has a preference for making most of the decisions itself. This will lead toward a high preference for microinvolvement. Management has a short-time horizon when making

decisions, which characterizes a high preference for microinvolvement. Since the management has a preference for being very involved in gathering and using detailed information when making decisions, a high preference for microinvolvement characterization is appropriate. The management of CMM Level 1 has a preference for wait and see and then act. This will lead toward a high preference for microinvolvement because management has to react to crisis at a very detailed level. Because CMM Level 1 is a small organization the preference for microinvolvement will be higher than it would otherwise be.

THE STRATEGY

The organization's strategy is categorized as one of either prospector, analyzer with innovation, analyzer without innovation, defender, or reactor. These categories follow Miles and Snow's typology. Based on your answers, the organization has been assigned to a strategy category. This is a statement of the current strategy; it is not an analysis of what is the best or preferred strategy for the organization.

Based on the answers you provided, it is most likely that your organization's strategy is a defender strategy (cf 70).
It could also be: an analyzer without innovation (cf 70).
An organization with a defender strategy is an organization that has a narrow product market domain. Top managers in this type of organization are expert in their organization's limited area of operation but do not tend to search outside their domains for new opportunities. As a result of this narrow focus, these organizations seldom need to make major adjustments in their technology, structure, or methods of operation. Instead, they devote primary attention to improving the efficiency of their existing operations.
CMM Level 1 has few products. It needs to defend these products well in the marketplace. Viability depends on being successful with these limited activities. When the top management of CMM Level 1 has a preference for a high level of microinvolvement, the strategy is likely to be defender.

An organization with an analyzer without innovation strategy is an organization whose goal is to move into new products or new markets only after their viability has been shown yet maintains an emphasis on its ongoing products. It has limited innovation related to the production process; generally an analyzer without innovation does not have product innovation.
The capital requirement of CMM Level 1 is not high, which is consistent with an analyzer without innovation strategy. With top management of CMM Level 1 preferring a high level of microinvolvement, the strategy is likely to be analyzer without innovation.

THE CURRENT ORGANIZATIONAL CHARACTERISTICS

Based on your answers, the organization's complexity, formalization, and centralization have been calculated. This is the current organization. Later in this report, there will be recommendations for the organization.

The current organizational complexity is low (cf 100).
The current horizontal differentiation is low (cf 100).
The current vertical differentiation is low (cf 100).
The current spatial differentiation is low (cf 100).
The current centralization is high (cf 100).
The current formalization is low (cf 100).
The current organization has been categorized with respect to formalization, centralization, and complexity. The categorization is based on the input you gave and does not take missing information into account.

SITUATION MISFITS

A situation misfit is an unbalanced situation among the contingency factors of management style, size, environment, technology, climate, and strategy.

The following misfits are present: (cf 100).

CMM Level 1 has a non routine technology, but the workforce has a low level of education and training. This situation can create production and service difficulties which usually require an investment in the education and training. A non routine technology usually requires that individuals adapt work methods to the particular task at hand. Individuals must have a sufficiently high level of skill to make these adaptations. Low levels of education and training do better at routine tasks and technologies. With a non routine technology and low level of education and training, new training will be required for the workforce. This training should emphasize individual responsibility and decision making for the quality of the product or service. Eg., it should provide new skills which permit the individual to take the imitative for action which meets the customers' requirements.

CMM Level 1 has a low product innovation but does not have a certain environment. This situation calls for a review and suggests that the organization consider greater product innovation. Low product innovation means the same products are available for an extended period. In a certain environment with little change in customer demands and preferences, there is little need for new products. But, with increasing uncertainty in customer demand, new competitor strategies, possible governmental actions, shifting customer tastes, etc., current products are likely to be mismatched with this changed environment. New products and innovation will likely be required to adapt and meet the emerging needs and opportunities of the new environment.

CMM Level 1 has both an analyzer strategy and a management with a short time horizon. Conflict and confusion are likely results. An analyzer is searching for opportunities which may not be within the current activities of the organization. Frequently, investment and startup costs will be incurred which will decrease short term returns. Management should develop a longer term outlook for the organization.

CMM Level 1 has both an analyzer strategy and few products. Generally, more products are required for an analyzer. A few products may be reasonable in the short run, but an analyzer should be in constant consideration of new possibilities. When a few, unchanging products become the norm, the analyzer should broaden its scope of new opportunities.

CMM Level 1 has both an analyzer without innovation strategy and an environment with high equivocality. A high equivocality in the environment calls for a capability to vary products and services as the environment becomes clear. Without an innovative capability, it may be very difficult to adjust. Copying what others have done maybe possible, but it is not likely to be viable for the long run.

CMM Level 1 has both an analyzer without innovation strategy and an uncertain environment. An uncertain environment calls for adaptability and change. Without innovation, the organization is limited to copy what others have done. It is inherently a risky position, but it may appear to be conservative. The organization needs to develop some innovative capabilities to adjust and adapt to the uncertainties in the environment.

CMM Level 1 has both an analyzer without innovation strategy and an environment with high or low complexity. A more innovative strategy is preferred. A highly complex environment involves a large number of variables which influence the organization.Without innovation, the organization is limited in its responses and its possible adaptations. A more innovative strategy is needed. In contrast, a low complexity environment has few variables to consider and may not provide enough potential for an analyzer to survive in the long run.

ORGANIZATIONAL CONSULTANT RECOMMENDATIONS

Based on your answers about the organization, its situation, and the conclusions with the greatest certainty factor from the analyses above Organizational Consultant has derived recommendations for the organization's configuration, complexity, formalization, and centralization. There are also recommendations for coordination and control, the appropriate media richness for communications, and incentives. More detailed recommendations for possible changes in the current organization are also provided.

ORGANIZATIONAL CONFIGURATIONS

The most likely configuration that best fits the situation has been estimated to be a simple configuration (cf 82).

It is certainly not: a machine bureaucracy (cf -100).

It is certainly not: a functional (cf -100).

It is certainly not: a professional bureaucracy (cf -100).

A simple organization has a flat hierarchy and a singular head for control and decision making.
Small organizations should very likely have a simple configuration. When the organization also has a nonroutine technology, the conclusion is even stronger. A nonroutine technology together with a desire from top management for a concentration of control make a simple configuration possible and likely. When the organization has an analyzer without innovation strategy, the conclusion is even stronger.

When the organization has a nonroutine technology, it is not likely that a machine bureaucracy is an efficient organization.
The configuration cannot be a functional configuration when the technology is nonroutine.

Because the organization does not have a routine technology, it is not likely that a professional bureaucracy is an efficient organization.
A professional bureaucracy is a less likely configuration when top management has a high preference for microinvolvement.

ORGANIZATIONAL CHARACTERISTICS

The recommended degree of organizational complexity is low (cf 69).
Small organizations should have low organizational complexity. Not much is known about the environment since both the environmental uncertainty and the environmental equivocality of CMM Level 1 are high. In this situation, the organizational complexity should be low. This allows the organization to adapt quickly. Top management of CMM Level 1 has a preference for a high level of microinvolvement, which leads to lower organizational complexity.

The recommended degree of horizontal differentiation is low (cf 69).
The recommended degree of vertical differentiation is low (cf 69).
The recommended degree of formalization is low (cf 58).
When the organization is in the service industry and it does not have a routine technology, its formalization should be lower than if it had been in the manufacturing industry. Organizations with nonroutine technology should have low formalization. Since the set of variables in the environment that will be important is not known and since it is not possible to predict what will happen, no efficient rules and procedures can be developed, which implies that CMM Level 1's formalization should be low.

The recommended degree of centralization is high (cf 42).
There is evidence against it should be: low (cf -17).
CMM Level 1 has an analyzer without innovation strategy. Centralization should be medium to high. There should be tight control over current activities and less control over new undertakings. When there is a medium capital requirement and the product innovation is low, as is the situation for CMM Level 1, centralization should be high to obtain efficiency. Small organizations should have a high degree of centralization. High centralization is required if top management has a preference for a high level of microinvolvement.
CMM Level 1's span of control should be narrow (cf 50).
Since CMM Level 1 has a nonroutine technology, it should have a narrow span of control.
CMM Level 1 should use media with high media richness (cf 91).
The information media that CMM Level 1 uses should provide a large amount of information (cf 91).
Incentives should be based on results (cf 91).
CMM Level 1 should use meetings as means for coordination and control (cf 91).

With a nonroutine technology CMM Level 1 should obtain coordination and control via group meetings. Media with high richness and large amount of information should be used. Incentives should be based on results.When the environment of CMM Level 1 has high equivocality, high uncertainty, and high complexity, coordination and control should be obtained through integrators and group meetings. The richness of the media should be high with a large amount of information. Incentives must be results based.Top management should play the central role in coordinating and controlling the activities of the organization as well as making strategic and operating decisions.
Top management should gather information, make decisions, and manage implementation. Top management should give direct orders to achieve the required coordination among the operations and activities.Top management should make many decisions. However, many individuals should be involved in gathering information and implementing those decisions.

ORGANIZATIONAL MISFITS

Organizational misfits compares the recommended organization with the current organization.
There are no organizational misfits (cf 100).

MORE DETAILED RECOMMENDATIONS

No detailed recommendations present (cf 100).
Based on the present input Organizational Consultant was not able to make any detailed recommendations.

END

REPORT SUMMARY - CMM Level 5

Time: 4:56:31 PM, 9/6/2000
Scenario: CMM_5

INPUT DATA SUMMARY

The description below summarizes and interprets your answers to the questions about your organization and its situation. It states your answers concerning the organization's current configuration, complexity, formalization, and centralization. Your responses to the various questions on the contingencies of age, size, technology, environment, management style, cultural climate and strategy factors are also given. The writeup below summarizes the input data for the analysis.

- CMM Level 5 has a professional bureaucracy configuration (cf 100).
- CMM Level 5 has a very large number of different jobs (cf 100).
- Of the employees at CMM Level 5 76 to 100 % have an advanced degree or many years of special training (cf 100).
- CMM Level 5 has 6 to 8 vertical levels separating top management from the bottom level of the organization (cf 100).
- The mean number of vertical levels is 6 to 8 (cf 100).
- CMM Level 5 has an undetermined number of separate geographic locations (cf 100).
- CMM Level 5's average distance of these separate units from the organization's headquarters is of no relevance because there is only one site undetermined (cf 100).
- 26 to 60 % of CMM Level 5's total workforce is located at these separate units (cf 100).
- Job descriptions are available for all employees, including senior management (cf 100).
- Where written job descriptions exist, the employees are supervised very closely to ensure compliance with standards set in the job description (cf 100).
- The employees are not allowed to deviate from the standards (cf 100).
- 81 to 100 % non-managerial employees are given written operating instructions or procedures for their job (cf 100).
- The written instructions or procedures given are followed to a very great extent (cf 100).
- Supervisors and middle managers are to a very great extent free from rules, procedures, and policies when they make decisions (cf 100).
- More than 80 % of all the rules and procedures that exist within the organization are in writing (cf 100).
- Top Management is to some extent involved in gathering the information they will use in making decisions (cf 100).
- Top management participates in the interpretation of an undetermined percentage of the information input (cf 100).
- Top management directly controls 41 to 60 % of the decisions executed (cf 100).
- The typical middle manager has some discretion over establishing his or her budget (cf 100).
- The typical middle manager has some discretion over how his/her unit will be evaluated (cf 100).
- The typical middle manager has some discretion over the hiring and firing of personnel (cf 100).
- The typical middle manager has some discretion over personnel rewards - (ie, salary increases and promotions) (cf 100).
- The typical middle manager has some discretion over purchasing equipment and supplies (cf 100).
- The typical middle manager has some discretion over establishing a new project or program (cf 100).
- The typical middle manager has great discretion over how work exceptions are to be handled (cf 100).
- CMM Level 5 has 50 employees (cf 100).
- CMM Level 5's age is mature (cf 100).
- CMM Level 5's ownership status is undetermined (cf 100).
- CMM Level 5 has many different products (cf 100).
- CMM Level 5 has many different markets (cf 100).
- CMM Level 5 operates at a high-activity level in more countries (cf 100).
- CMM Level 5 has many different products in the foreign markets (cf 100).
- CMM Level 5's major activity is categorized as service (cf 100).
- CMM Level 5 has a specialized customer-oriented service technology (cf 100).

- CMM Level 5 has a routine technology (cf 100).
- CMM Level 5's technology is undetermined with respect to divisibility (cf 100).
- CMM Level 5's technology dominance is strong (cf 100).
- CMM Level 5 has either planned or already has an advanced information system (cf 100).
- CMM Level 5's environment is complex (cf 100).
- The uncertainty of CMM Level 5's environment is high (cf 100).
- The equivocality of the organization's environment is high (cf 100).
- CMM Level 5's environment has an undetermined level of hostility (cf 100).
- Top management prefers to make policy and general resource allocation decisions (cf 100).
- Top management primarily prefers to make long-term decisions (cf 100).
- Top management has a preference for very aggregate information when making decisions (cf 100).
- Top management has a preference for proactive actions (cf 100).
- Top management is risk neutral (cf 100).
- Top management has a preference for a combination of motivation and control (cf 100).
- CMM Level 5 operates in an industry with a high capital requirement (cf 100).
- CMM Level 5 has a high product innovation (cf 100).
- CMM Level 5 has a high process innovation (cf 100).
- CMM Level 5 has a high concern for quality (cf 100).
- CMM Level 5's price level is undetermined relative to its competitors (cf 100).
- The level of trust is not known (cf 100).
- The level of conflict is not known (cf 100).
- The employee morale is not known (cf 100).
- Rewards are given in a not known fashion (cf 100).
- The resistance to change is not known (cf 100).
- The leader credibility is not known (cf 100).
- The level of scapegoating is not known (cf 100).

THE SIZE

The size of the organization - large, medium, or small - is based upon the number of employees, adjusted for their level of education or technical skills.

Based on the answers you provided, it is most likely that your organization's size is medium (cf 50).

More than 75 % of the people employed by CMM Level 5 have a high level of education. Adjustments are made to this effect. The adjusted number of employees is lower than 500 but greater than 100 and CMM Level 5 is categorized as medium. However, for this adjusted number this size does not have a major effect on the organizational structure.

THE CLIMATE

The organizational climate effect is the summary measure of people and behavior.

Based on the answers you provided, it is most likely that the organizational climate is a unknown climate (cf 100).

THE MANAGEMENT STYLE

The level of management's microinvolvement in decision making is the summary measure of management style. Leaders have a low preference for microinvolvement; managers have a high preference for microinvolvement.

Based on the answers you provided, it is most likely that your management profile has a low preference for microinvolvement (cf 81).

The management of CMM Level 5 has a preference for delegating decisions. This will lead toward a low preference for microinvolvement. Management has a long-term horizon when making decisions, which characterizes a preference for a low microinvolvement. Since the management has a preference for making decisions on the basis of very aggregate information a low preference for microinvolvement characterization is appropriate. The management of CMM Level 5 has a preference for taking actions when making decisions. This will lead toward a low preference for microinvolvement because meeting the problems before they arise allow you to work on the general level and not being consumed with the very detailed decisions that can best be made at lower level in the organization.

THE STRATEGY

The organization's strategy is categorized as one of either prospector, analyzer with innovation, analyzer without innovation, defender, or reactor. These categories follow Miles and Snow's typology. Based on your answers, the organization has been assigned to a strategy category. This is a statement of the current strategy; it is not an analysis of what is the best or preferred strategy for the organization.

Based on the answers you provided, it is most likely that your organization's strategy is a prospector strategy (cf 84).
It could also be: a defender (cf 81).
An organization with a prospector strategy is an organization that continually searches for market opportunities and regularly experiments with potential responses to emerging environmental trends. Thus, the organization is often the creator of change and uncertainty to which its competitors must respond. However, because of its strong concern for product and market innovation, a prospector usually is not completely efficient.
CMM Level 5 has numerous products. A prospector is constantly seeking new product opportunities to serve the existing and potentially new customers. For a prospector strategy to be aggressive in product development or market opportunities exploitation, it requires a high capital investment. With a concern for high quality a prospector strategy is a likely strategy for CMM Level 5. With top management preferring a relatively low level of microinvolvement, the strategy is likely to be prospector.

An organization with a defender strategy is an organization that has a narrow product market domain. Top managers in this type of organization are expert in their organization's limited area of operation but do not tend to search outside their domains for new opportunities. As a result of this narrow focus, these organizations seldom need to make major adjustments in their technology, structure, or methods of operation. Instead, they devote primary attention to improving the efficiency of their existing operations.
For a company with a high capital investment, the ability to adjust its capital base quickly is not likely. Thus, it needs to protect and defend its position; a defender strategy and technology protection is appropriate. CMM Level 5 has a routine technology. Consequently, new products for new customers are less likely to be possible. It needs to defend its position for the technology it has or copy well-known products or markets. With a concern for high quality a defender strategy is a likely strategy for CMM Level 5.

THE CURRENT ORGANIZATIONAL CHARACTERISTICS

Based on your answers, the organization's complexity, formalization, and centralization have been calculated. This is the current organization. Later in this report, there will be recommendations for the organization.

The current organizational complexity is high (cf 100).
The current horizontal differentiation is high (cf 100).
The current vertical differentiation is medium (cf 100).
The current spatial differentiation is medium (cf 100).
The current centralization is medium (cf 100).
The current formalization is high (cf 100).
The current organization has been categorized with respect to formalization, centralization, and complexity. The categorization is based on the input you gave and does not take missing information into account.

SITUATION MISFITS

A situation misfit is an unbalanced situation among the contingency factors of management style, size, environment, technology, climate, and strategy.

The following misfits are present: (cf 100).

CMM Level 5 has a routine technology but it does not have a certain environment. This situation can cause problems for which a more non routine technology is better! A routine technology produces goods and services efficiently which are standard and without variation. In an uncertain environment, it is very likely that the customers will prefer variation in products and services. Competitors are likely to vary their strategies in products, prices, advertising, etc. New innovative strategies may be called for. A more non routine technology will likely be required to adapt to an uncertain environment.
CMM Level 5 has both a routine technology and a highly equivocal environment. A more non routine technology is a better fit with an equivocal environment. A routine technology produces goods and services efficiently which are standard and without variation. In a highly equivocal environment, it is likely that customers will demand variation in the product and service characteristics. Competitors are likely to introduce new products, vary prices, modify advertising, etc. Further, in the equivocal environment, large changes can come from unforeseen actions by competitors, government, and breakthrough innovations. A more non routine technology will be required to adapt to the unknowns and changes of an equivocal environment.
CMM Level 5 has both a routine technology and a high requirement for product innovation. This situation must be changed; a routine technology will not support high product innovation. A routine technology yields standard products with low variation. The need for product innovation creates a mismatch. Product innovation will be difficult to manage, expensive and inefficient. For product innovation, a more non routine and adaptable technology is required. Of course, the organization may also shift to markets and products where less product innovation is required and a routine technology is suitable.
CMM Level 5 has both a prospector strategy and routine technology. These are not compatible. A prospector innovates with new ideas and products. With a routine technology, developing new products or services will be very difficult. The routine technology is limited in its capacity to vary products or processes. For a prospector, a more non routine technology is required.
CMM Level 5 has a high capital requirement but is not a large organization. The organization can be vulnerable. An organization with a high capital requirement and a few employees usually makes a few standardized products. Further, the technology is likely to be very limited in adaptiveness. The organization is then vulnerable to changes in the environment, market and products changes. Smaller organizations with small capital requirements are frequently more adaptive. To reduce this vulnerability, the organization should consider creating a greater capability for adaptation, which will usually require more employees of higher skill, education and training.

ORGANIZATIONAL CONSULTANT RECOMMENDATIONS

Based on your answers about the organization, its situation, and the conclusions with the greatest certainty factor from the analyses above Organizational Consultant has derived recommendations for the organization's configuration, complexity, formalization, and centralization. There are also recommendations for coordination and control, the appropriate media richness for communications, and incentives. More detailed recommendations for possible changes in the current organization are also provided.

ORGANIZATIONAL CONFIGURATIONS

The most likely configuration that best fits the situation has been estimated to be a matrix configuration (cf 53).
It could also be: a divisional (cf 52).
It is certainly not: a machine bureaucracy (cf -84).
It is certainly not: a professional bureaucracy (cf -84).

It is certainly not: an adhocracy (cf -100).
A matrix structure is a structure that assigns specialists from functional departments to work on one or more interdisciplinary teams that are led by project leaders. Permanent product teams are also possible. A dual hierarchy manages the same activities and individuals at the same time.
When CMM Level 5 has many products or markets, a matrix configuration is a likely configuration. When CMM Level 5's environment has neither low equivocality nor low complexity, the configuration should be matrix. When CMM Level 5 is of medium size, the configuration can be a matrix configuration. The matrix configuration is a more likely configuration when CMM Level 5 has a unit production technology. When strategy is prospector, the organizational configuration can be a matrix configuration.
The matrix structure may be a trans-national structure. When foreign product/service diversity is high and international involvement is high, CMM Level 5 should have a multidimensional, global configuration.

The matrix configuration is usually not a very efficient configuration when the technology is a routine technology.
A divisional organization is an organization with self-contained unit grouping into relatively autonomous units coordinated by a headquarters, (product, customer, or geographical grouping).
When the organization is of medium size, the configuration can be a divisional configuration. Because the organization has many products, the configuration should be divisional.

When the organization has a prospector strategy, it cannot be a machine bureaucracy!
Since the organization has a prospector strategy, it cannot have a configuration like a professional bureaucracy.
When the technology is very routine, the configuration cannot be an ad hoc configuration because it will not be able to operate!

ORGANIZATIONAL CHARACTERISTICS

The recommended degree of organizational complexity is low (cf 40).
CMM Level 5 has a prospector strategy. Then, the organizational complexity should be either low or high. CMM Level 5 has a routine technology, which implies that the organizational complexity should be low. Not much is known about the environment since both the environmental uncertainty and the environmental equivocality of CMM Level 5 are high. In this situation, the organizational complexity should be low. This allows the organization to adapt quickly.

Medium size organizations should have medium organizational complexity. Because CMM Level 5 has an advanced information system, organizational complexity can be greater than it could otherwise.

CMM Level 5 has a prospector strategy. Then, the organizational complexity should be either low or high. Top management of CMM Level 5 has a preference for a low level of microinvolvement, which allows for a higher organizational complexity. Because CMM Level 5 has an advanced information system, organizational complexity can be greater than it could otherwise.

The recommended degree of horizontal differentiation is high (cf 49).
The recommended degree of vertical differentiation is low (cf 70).
The recommended degree of formalization is low (cf 53).

CMM Level 5 has a prospector strategy. A low formalization is required so that the organization can react quickly. Low formalization is also required because of the need for innovations. Since the set of variables in the environment that will be important is not known and since it is not possible to predict what will happen, no efficient rules and procedures can be developed, which implies that CMM Level 5's formalization should be low. Low formalization is consistent with top management having a low preference for microinvolvement.

The recommended degree of centralization is medium (cf 27).

When there is a high capital requirement and the product innovation is high, as is the case for CMM Level 5, centralization should be medium. CMM Level 5 is of medium size. Such organizations should have medium to high centralization. Because CMM Level 5 has an advanced information system, centralization can be greater than it could otherwise.

CMM Level 5 has a prospector strategy. A low centralization is required so that the organization can react and innovate quickly. Low centralization can be allowed when top management has no desire for microinvolvement. Since there are many factors in the environment that affect the organization but CMM Level 5 does not know which factors are or will be important for CMM Level 5, centralization should be low.

CMM Level 5's span of control should be wide (cf 50).
Since CMM Level 5 has a routine technology, it should have a wide span of control.
CMM Level 5 should use media with high media richness (cf 70).
It also should use media with low media richness (cf 70).

The information media that CMM Level 5 uses should provide a large amount of information (cf 70).
The media used should also provide a small amount of information (cf 70).
Incentives should be based on results (cf 70).
It should also be based on procedures (cf 70).
CMM Level 5 should use meetings as means for coordination and control (cf 70).
It should also use integrators (cf 70).
Since CMM Level 5 is not small and has a routine technology, coordination and control should be obtained via rules and planning, and media with low richness and a small amount of information can be used. Incentives should be based on process.When the environment of CMM Level 5 has high equivocality, high uncertainty, and high complexity, coordination and control should be obtained through integrators and group meetings. The richness of the media should be high with a large amount of information. Incentives must be results based.Coordination is a major issue requiring a lot of time by functional managers and product or project managers. Managers should make frequent adjustments in order to maintain project and product goals and use scarce functional resources and personnel efficiently. In an international firm, matrix dimensions will likely include country or region and may include either product, customer, or function.Project or product managers will likely be required to champion new innovations in customers, products or technologies.

ORGANIZATIONAL MISFITS

Organizational misfits compares the recommended organization with the current organization.
The following organizational misfits are present: (cf 100).
Current and prescribed configuration do not match.
Current and prescribed complexity do not match.
Current and prescribed formalization do not match.

MORE DETAILED RECOMMENDATIONS

There are a number of more detailed recommendations (cf 100).
You may consider decreasing the number of positions for which job descriptions are available.
You may consider supervising the employees less closely.
You may consider allowing employees more latitude from standards.
You may consider fewer written job descriptions.
Managerial employees may be asked to pay less attention to written instructions and procedures.
You may consider having fewer rules and procedures put in writing.

END

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B
SIMULATION REPORTS

The following chart (Fig B.1) presents the simulated organization and the simulated software process. The process presents only four cycles of evolution. Each cycle has the activities described in Chapter VII (Fig. 7.1).

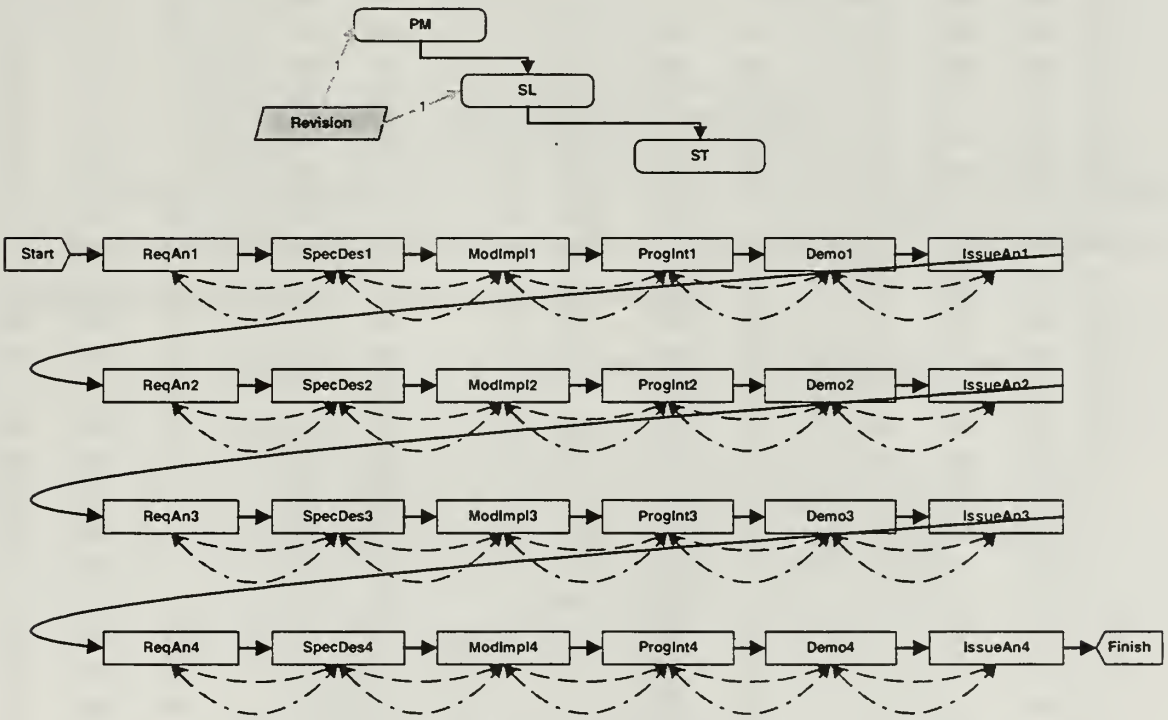


Figure B.1: Project Layout[1]

[1] Note: The detailed description of the notation can be found on the VitéProject user manual (Levitt, 1999). Rectangles indicate tasks. Rounded-corner rectangles indicate roles. Parallelograms indicate meetings. Double-headed-dashed arrows indicate information dependencies between tasks. Dashed arrows indicate problem dependencies between tasks. Normal arrows indicate precedence dependencies between tasks.

1. Simulation Results

Table B.1 shows the estimated duration of the project for different scenarios. There are 30 data points for each scenario.

Table B.1: Simulation Results

LLL	LLH	LLH(2.5)	LLH(5)	LHL	LHH	LHH(2.5)	LHH(5)	HLL	HLH	HLH(2.5)	HLH(5)	HHL	HHH	HHH(2.5)	HHH(5)
78	91	228	455	91	108	270	540	29	37	93	185	37	43	108	215
80	91	228	455	91	112	280	560	29	38	95	190	38	44	110	220
81	93	233	465	91	115	288	575	30	38	95	190	38	45	113	225
82	94	235	470	92	115	288	575	30	38	95	190	38	45	113	225
82	94	235	470	93	118	295	590	30	39	98	195	39	45	113	225
82	95	238	475	94	118	295	590	31	39	98	195	39	46	115	230
83	95	238	475	95	120	300	600	31	40	100	200	39	46	115	230
85	96	240	480	96	122	305	610	31	40	100	200	39	46	115	230
85	97	243	485	96	123	308	615	31	40	100	200	39	46	115	230
86	98	245	490	96	123	308	615	31	40	100	200	40	46	115	230
87	98	245	490	96	124	310	620	31	40	100	200	40	46	115	230
88	99	248	495	96	124	310	620	31	40	100	200	40	47	118	235
88	100	250	500	98	125	313	625	32	41	103	205	41	48	120	240
88	100	250	500	100	126	315	630	32	41	103	205	41	48	120	240
88	101	253	505	100	127	318	635	32	41	103	205	41	48	120	240
88	102	255	510	101	127	318	635	32	41	103	205	42	48	120	240
89	102	255	510	101	128	320	640	32	42	105	210	42	48	120	240
89	103	258	515	102	129	323	645	33	42	105	210	42	49	123	245
90	104	260	520	102	129	323	645	33	43	108	215	43	49	123	245
90	104	260	520	103	130	325	650	33	43	108	215	43	49	123	245
90	107	268	535	104	131	328	655	33	44	110	220	43	50	125	250
90	107	268	535	104	132	330	660	33	44	110	220	43	51	128	255
91	107	268	535	107	134	335	670	34	45	113	225	44	51	128	255
91	107	268	535	107	137	343	685	34	45	113	225	44	52	130	260
92	107	268	535	108	138	345	690	34	45	113	225	44	54	135	270
92	109	273	545	109	139	348	695	35	45	113	225	44	54	135	270
93	110	275	550	110	139	348	695	35	45	113	225	45	54	135	270
95	110	275	550	111	142	355	710	35	45	113	225	46	54	135	270
97	111	278	555	113	143	358	715	35	46	115	230	46	55	138	275
100	112	280	560	119	150	375	750	35	47	118	235	47	57	143	285

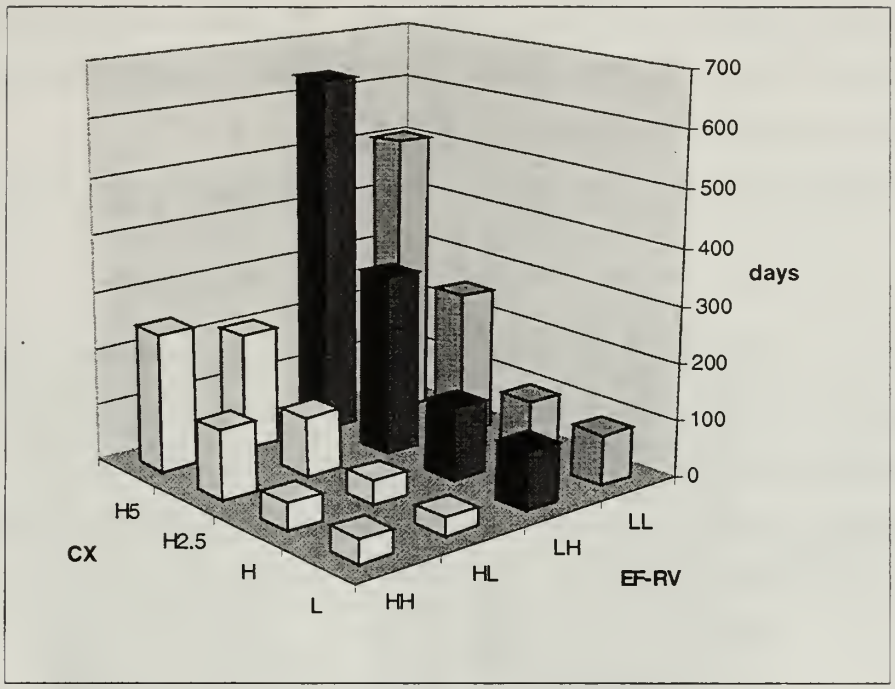


Figure B.2: Effects of Complexity

Figure B.2 shows the effects of the complexity in the expected duration of similar efficiency and requirements volatility scenarios. Observe that the effect of complexity is different when the efficiency and requirement volatility vary.

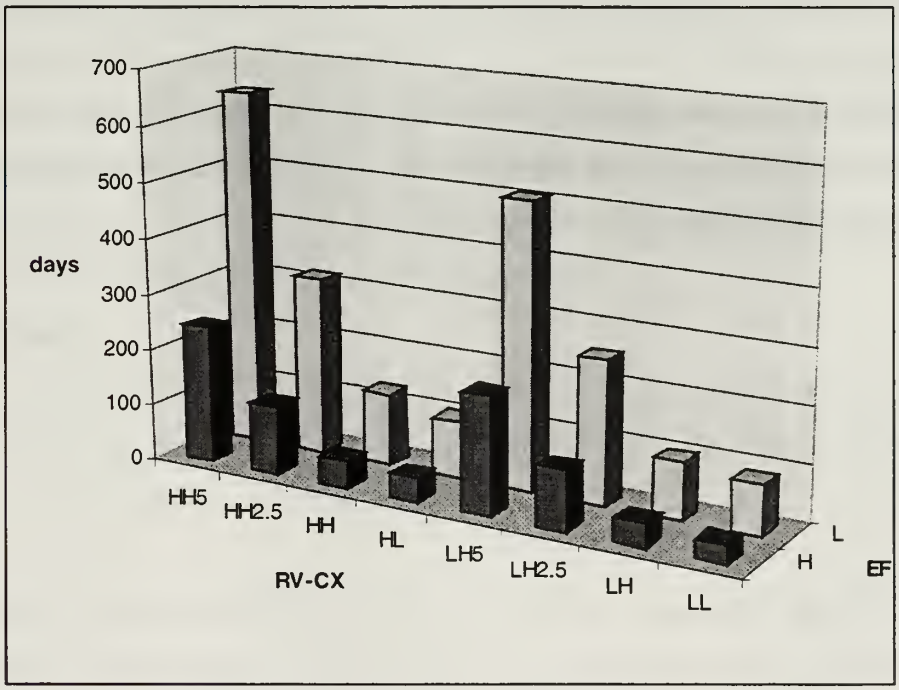


Figure B.3: Effects of Efficiency

Figure B.3 shows the effects of efficiency. For same values of complexity and requirements volatility, the durations for high efficiency scenarios were 40% of the durations for low efficiency ones.

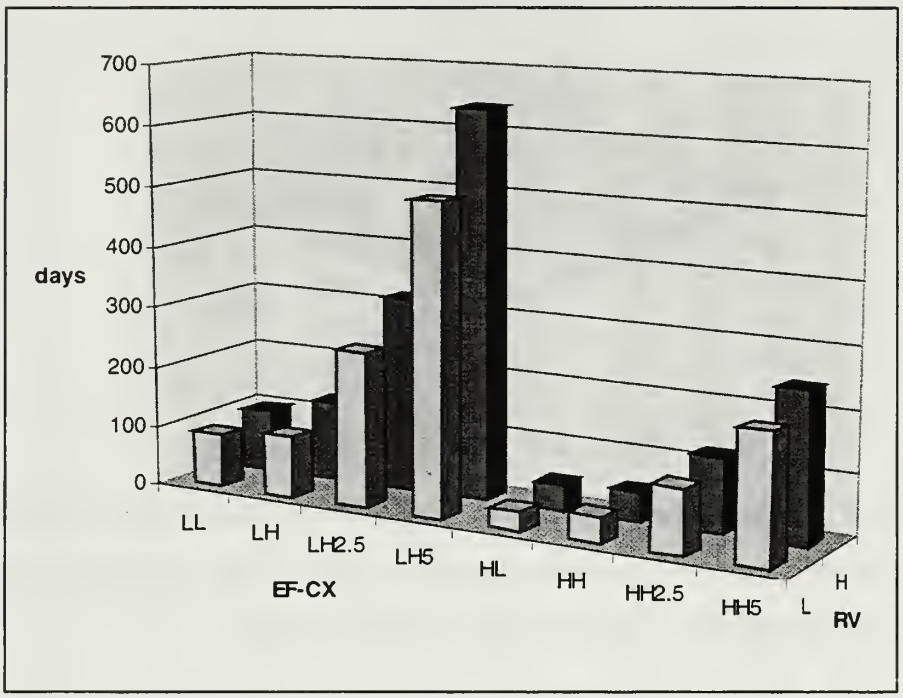


Figure B.4: Effects of Requirements Volatility

Figure B.4 shows the effects of requirements volatility. For same values of complexity and efficiency, the durations for volatile scenarios were 122% of the durations for non-volatile ones.

APPENDIX C
PARAMETER CONFIGURATION FOR VITEPROJECT

VitéProject uses a set of default values for the variables of the model. These values are stored in a file named "behmatrx.opd" in the subdirectory of VitéProject. The behavior of the model depends on the values of these variables that are collectively called Behavior Matrix. This Appendix discusses the concepts considered in the behavior matrix and their relationship with software projects. The simulations used the default values for this file.

(1) Participant attention rule: Defines the probability distribution applied to the different selection methods (e.g. priority, FIFO, LIFO, random) of picking items to process.

(2) Participant tool selection rules: Defines the probability distribution applied to different information exchange tools (e.g. conversation, email, fax, memo, phone, video, voice-mail) given the type of message (e.g. Exception, Decision, etc.) A tool selected for an information exchange determines (1) the time needed for the message to move from one participant to another and (2) the time the message will stay in the in-tray of the receiver participant.

Findings:

i. Even if there is one matrix for each role, all the matrices are identical.

ii. Too much emphasis on voicemail. We expected more weight on conversation, phone and email.

(3) Activity Verification Failure Probability (VFP) adjustment: There are two VFP (internal and external). The internal VFP depends on the complexity of the requirement and the skills of the participants. The external VFP depends on the

complexity of the solution and the skills of the participants. The processing speed of responsible participants is affected by the solution complexity and the requirement complexity.

(4) Activity Information Exchange Frequency adjustment: This adjustment depends on the uncertainty of the activity and the team experience.

(5) Participant Processing Speed adjustment: This adjustment depends on the match between the participant and activity skill requirements.

(6) Definition of Rework, Quick-Fix, and Ignore decisions: This matrix defines how much of the original failed work should be reworked, quick-fixed or ignored. The values depend on the following failure types:

i. InternalIInternal: Amount of rework of an activity given internal activity failure (based on VFPInternal.).

ii. InternalIExternal: Amount of rework of an activity given external failure (based on VFP External.).

iii. ExternalIExternal: Amount of rework of a failure dependent activity given external failure of an independent activity (based on VFP External of the independent activity.).

(7) Impact of participant information exchange behavior on its VFP: This adjustment depends on the attendance or non-attendance of the participant to information exchange events related to the activities.

(8) Impact of participant decision-making behavior on the VFP of failed activity: This adjustment depends on the centralization level of the organization.

(9) The probabilities used by VitéProject were set as follows:

- *Functional Error Rate* (0.01 low). Functional errors is the number of generated internal functional errors, shown in the Simulator Analysis Summary.
- *Project Error Rate* (0.01 low). Project errors is the number of generated project errors, shown in the Simulator Analysis Summary.
- *Information Exchange* (0.8 high)
- *Noise* (0.1 normal)

(10) Finally there is a set of matrices to implement Project Decision Making Policies including how to determine to whom to report an exception, how to make a decision for an exception, what is the maximum time a participant will wait before it takes delegation by default.

The following source code is the behavior matrix provided as default by VitеProject.

```
%============================================================================%
%
%
% BehMatrx.opd - Vité-Project uses default qualitative-to-quantitative calibration
% parameter values defined in this file.  To override any of the default calibration
% values, place a modified version of this file in the directory that holds Vité-Project
% and specify the file name in the Vité-Project simulation control dialog box.  Vité-
% Project will load this file automatically.
%
% Each matrix defines an association set: the row selection, when associated with the
% column selection, has the behavior of the corresponding matrix value.  For example, for
% the ParticipantAttentionRule, a Project Manager (PM) will select an item from the
% intray by Priority with probability 0.5.  Notation:
% PM = Project Manager
% SL = participant subteam leader
% ST = participant
%
% Revisions:
%  10.17.97 Update comments and values
%============================================================================%

(Application BehMatrices)

%============================================================================
% Participant attention rule: - A participant uses this attention rule to select an item
% from its in-tray.  By default, all participants in Vité-Project share this common
% attention rule.
% Example: a Project Manager (PM) will select an item from the intray by priority with
% probability 0.5, with FIFO with probability 0.1, etc.
%============================================================================

(Matrix ParticipantAttentionRule
    :Row       PM SL ST                          %= Participant role.
    :Column    Priority FIFO LIFO Random     %= Item Selection strategy.
    :Values    (0.4 0.3 0.2 0.1)             %= Probability corresponding strategy
           (0.3 0.4  0.2  0.1)               % will be applied.
              (0.1 0.5  0.3  0.1)
```

```
)

%=============================================================================
%  Participant tool selection rules
%  Information exchange tool selection is based on only Message types (e.g,. Exception,
%  Decision, etc.)  A tool selected for an information exchange determines (1) the time
%  needed for the message to move from one participant to another and (2) the time the
%  message will stay in the in-tray of the receiver participant.
%  Example: Given an exception to process, the PM will never choose the Phone or Video.
%  Note that Decisions go directly to the recipient in-tray without use of a information
%  exchange tool.
%=============================================================================

%-----------------------------------------------------------------------
%  This rule only applies to project managers
%
(Matrix ToolSelectionRulesPM
    :Row [Message type]: Decision Exception InfoExchange    Meeting Noise
    :Column    [Tool to use]: Conversation Email Fax Memo Phone Video VoiceMail
    :Values    (0.15 0.20 0.20 0.20 0.0  0.0 0.25)   %= Probability
               (0.20 0.20 0.20 0.20 0.1  0.0 0.10)   % a specific tool
               (0.25 0.1  0.1  0.15 0.25 0.0 0.15)   % will be used
               (0.5  0.0  0.0  0.0  0.2  0.0 0.3)
               (0.3  0.1  0.05 0.1  0.35 0.0 0.1)
)
%-----------------------------------------------------------------------
%  This rule only applies to participant leaders
%
(Matrix ToolSelectionRulesSL
    :Row       Decision Exception InfoExchange Meeting Noise
    :Column    Conversation Email Fax Memo Phone Video VoiceMail
    :Values    (0.15 0.20 0.20 0.20 0.0  0.0 0.25)   %= Probability
               (0.20 0.20 0.20 0.20 0.1  0.0 0.10)   % a specific tool
               (0.25 0.1  0.1  0.15 0.25 0.0 0.15)   % will be used
               (0.5  0.0  0.0  0.0  0.2  0.0 0.3)
               (0.3  0.1  0.05 0.1  0.35 0.0 0.1)
)
%-----------------------------------------------------------------------
%  This rule only applies to sub teams
%
(Matrix ToolSelectionRulesST
    :Row       Decision Exception InfoExchange Meeting Noise
    :Column    Conversation Email Fax Memo Phone Video VoiceMail
    :Values    (0.15 0.20 0.20 0.20 0.0  0.0 0.25)   %= Probability
               (0.20 0.20 0.20 0.20 0.1  0.0 0.10)   % a specific tool
               (0.25 0.1  0.10 0.15 0.25 0.0 0.15)   % will be used
               (0.5  0.0  0.0  0.0  0.2  0.0 0.3)
               (0.3  0.1  0.05 0.10 0.35 0.0 0.10)
)

%=============================================================================
%  Activity Verification Failure Probability (VFP) adjustment:
%     The formula used to determine activities' internal and external VFP:
%
% ?activity.VFPexternal =
%    ?proj.VFPexternal * SolutionComplexityEffect * ParticipantSkillEffect;
% ?activity.VFPinternal =
%     ?proj.VFPinternal * RequirementComplexityEffect * ParticipantSkillEffect;
%
%     The adjustment coefficients (e.g., SolutionComplexityEffect ParticipantSkillEffect)
% are determined by values in the following matrices.

%=============================================================================

%-----------------------------------------------------------------------
%  Effect of Activity solution complexity on processing speed of responsible
%  participants.
%
(Matrix SolutionComplexityEffect
    :Row       High Medium Low       %= Level of solution complexity.
    :Values    1.5 1.0 0.67   %= Value of SolutionComplexityEffect
```

```
)

%-----------------------------------------------------------------
%  Effect of Activity requirement complexity on responsible participant processing speed.
%
(Matrix RequirementComplexityEffect
    :Row       High Medium Low       %= Level of requirement complexity.
    :Values    1.5 1.0 0.67          %= Value of RequirementComplexityEffect
)

%-----------------------------------------------------------------
%  Effect of Participant-Activity skill match on activity VFP:
%  If responsible participant skill matches the skill requirement of the
%  corresponding activity, then use this matrix to determine
%  ParticipantSkillEffect.
%
(Matrix ParticipantSkillMatchVFP
    :Row       High Medium Low       %= Level of participant App. Experience
    :Column    High Medium Low       %= Participant Required Skill Level.
    :Values    (0.5 0.7 0.9)         %= Values of ParticipantSkillEffect.
               (0.7 1.0 1.2)
               (0.9 1.2 1.5)
)

%-----------------------------------------------------------------
%  Effect of Participant-Activity match on activity VFP:
%  If participant skill DOES NOT match activity's skill requirement, then
%  use this matrix to determine ParticipantSkillEffect.  Failure of
%  participant-activity skill match creates a major VFP penalty.
%
(Matrix ParticipantSkillNonMatchVFP
    :Row       High Medium Low       %= Level of participant App. Experience
    :Column    High Medium Low       %= Participant other Skill Level.
    :Values    (2.0 2.0 2.0)         %= Values of ParticipantSkillEffect.
               (2.5 2.5 2.5)
               (3.5 3.5 3.5)
)

%=================================================================
%  Activity Information Exchange Frequency adjustment: The following formula is used to
%  determine probabilistic information exchange frequency of an activity
%
% ?activity.InfoExchangeFrequency = ?proj.InfoExchangeFrequency *
%  ActivityUncertaintyEffect * TeamExperienceEffect
%=================================================================

%-----------------------------------------------------------------
%  Effect of Activity uncertainty on information exchange frequency:
%
(Matrix ActivityUncertaintyEffect
    :Row       High Medium Low       %= Level of activity uncertainty
    :Values    1.4 1.00 0.67  %= Value of ActivityUncertaintyEffect
)

%-----------------------------------------------------------------
%  Effect of responsible Participant team experience on information exchange frequency:
%
(Matrix TeamExperienceEffect
    :Row       High Medium Low       %= Level of participant team experience.
    :Values    0.67 1.0 1.5   %= Value of TeamExperienceEffect
)

%=================================================================
%  participant processing speed adjustment:
%     The following formula determines participant processing speed.  Since participant
%     processing speed is based on its match with the skill requirement of its assigned
%     activity, the ParticipantSpeed is associated with each activity.  (Vité-Project
%     assumes that each activity can have only ONE responsible participant working on
%     it.)
%
% ?activity.ResponsibleParticipantSpeed =
```

```
%          1.0 / (?Participant.NumberOfParticipants * ?Participant.APS0 *
% ParticipantSkillEffect * ?Participant.TimePercentageForProject);
%
%     The rule uses 1/ "time needed to process a work unit" to calculate speed.
%=============================================================================

%-----------------------------------------------------------------------
%  Effect of Participant-Activity match on APS:
%  If responsible participant skill matches the activity's skill
%  requirement, then use this matrix to determine the value of
%  ParticipantSkillEffect.
%
 (Matrix ParticipantSkillMatchAPS
    :Row       High Medium Low       %= Level of participant App.Experience.
    :Column    High Medium Low       %= Participant Required Skill level.
    :Values    (2.0 1.5 0.9)         %= Values of ParticipantSkillEffect
               (1.5 1.0 0.7)
               (0.9 0.7 0.5)
)

%-----------------------------------------------------------------------
%  If participant skill DOES NOT match activity's skill requirement, then
%  use this matrix to determine the value of ParticipantSkillEffect.
%
(Matrix ParticipantSkillNonMatchAPS
    :Row       High Medium Low       %= Level of participant App. Experience.
    :Column    High Medium Low       %= Participant Other Skill level.
    :Values    (0.7 0.7 0.7)         %= Values of ParticipantSkillEffect
               (0.5 0.5 0.5)
               (0.3 0.3 0.3)
)

%=============================================================================
%  Definition of Rework, Quick-Fix, and Ignore decisions:
%  This matrix defines how much of the original failed work should be reworked based
%  decision types (i.e., Reworked, Quick-Fixed, Ignore).  The actual rework volume is the
%  given subactivity volume * % - of failed work that needs to be reworked * user-
%  interface defined "Strength" of failure dependent activity relationship
%
%    The values change depending on the failure types described below:
%
%    Internal!Internal: Amount of rework of an activity given internal activity failure
%    (based on VFPInternal.)
%
%    Internal!External: Amount of rework of an activity given external failure (based on
%    VFP External.)
%    Internal!External Amount of rework of a failure dependent activity given external
%    failure of an independent activity (based on VFP External of the independent
%    activity.)
%
%=============================================================================
%
(Matrix ReworkVolume
    :Row       Internal Internal!External External!External %= failure type
    :Column    Rework Quick-Fix Ignore       %= Decision for the exception
    :Values    (1.0 0.5 0.0)                 %= Percent of failed work
               (1.0 0.5 0.0)                 % that needs to be reworked.
               (1.0 0.5 0.0)
)

%=============================================================================
%  Impact of participant information exchange behavior on its VFP:
%  Vité-Project simulates the impact of participant information exchange behavior on its
%  VFP by updating VFP based on the effect weight as shown below (same for VFPexternal
%  and VFPinternal):
%
%  ?activity.VFPinternal = ?activity.VFPinternal * VFPInfoXEffect;
%  if   ?activity.VFPinternal > 1.0;
%  then ?activity.VFPinternal = 1.0;
%
%  The value of VFPInfoXEffect is retrieved from the following matrices.
```

```
%
%  VFP updating is dynamic, i.e., it happens whenever an information exchange finishes.
%  You can disable the effects by setting matrix values to 1.0.
%=============================================================================

%-----------------------------------------------------------------
%  This matrix defines the weight for updating participant verification failure
probabilities (*** Internal and External) due to not attending to information exchange
with peers, meetings and noise respectively.
%  NOTE: Weight =1.0 implies no effect of ignoring communications, meetings or noise.
%
(Matrix ParticipantNonAttendanceFailureEffect
    :Row       InfoXNonAttend MeetNonAttend NoiseNonAttend
                                        %= Nonatt InfoX type
    :Column    High Medium Low              %= Level of formalization
    :Values    (1.01 1.07 1.1)              %= VFPInfoXEffect.
            (1.10 1.07 1.05)
               (1.0  1.00 1.00)
)

%-----------------------------------------------------------------
%  This matrix defines the weight for updating participant verification failure
%  probability due to attending to information exchange from peers, meetings
%  and noise respectively.
%

(Matrix ParticipantAttendanceFailureEffect
    :Row       InfoXAttend MeetAttend NoiseAttend %= Nonatt InfoX type
    :Column    High Medium Low                       %= Level of formalization
    :Values    (0.99  0.96  0.95)                 %= VFPInfoXEffect.
            (0.90  0.96  0.99)
               (1.0   1.0   1.0)
)

%=============================================================================
%  Impact of participant decision-making behavior on the VFP of failed activity:
%  Vité-Project simulates the impact of participant information exchange behavior on its
%  VFP  updating VFP based on the effect weight as shown below
%  (same for VFPexternal):
%
%        ?activity.VFPinternal = ?activity.VFPinternal * VFPInfoXEffect;
%        if   ?activity.VFPinternal > 1.0;
%        then ?activity.VFPinternal = 1.0;
%
%     The value of VFPInfoXEffect is retrieved from the following matrices, based
%     decision-maker's role and the type of decision it has made.
%
%     VFP updating is dynamic, i.e., it happens whenever a decision is made.
%
%     You can turn off the effects by setting values of the matrices to 1.0.
%=============================================================================

%-----------------------------------------------------------------
%  Matrix used for Low centralization:
%
(Matrix LowCentralDecisionWeight
    :Row       PM SL ST                     %= Decision-maker's role.
    :Column    Rework Quick-Fix Ignore %= Type of decision made.
    :Values    (0.95 1.0 1.05)              %= VFPInfoXEffect for update VFP
               (0.95 1.0 1.05)
               (0.95 1.0 1.05)
)

%-----------------------------------------------------------------
%  Matrix used for Medium centralization:
%
(Matrix MediumCentralDecisionWeight
    :Row       PM SL ST                     %= Decision-maker's role.
    :Column    Rework Quick-Fix Ignore %= Type of decision made.
    :Values    (0.9  0.95 1.05)             %= VFPInfoXEffect for update VFP
               (0.95 1.0 1.05)
```

```
              (0.95 1.05 1.1)
)

%-------------------------------------------------------------------------
%  Matrix used for High centralization:
%
(Matrix HighCentralDecisionWeight
    :Row       PM SL ST                        %= Decision-maker's role.
    :Column    Rework Quick-Fix Ignore %= Type of decision made.
    :Values    (0.9  0.95 1.05)                %= VFPInfoXEffect for update VFP
               (0.95  1.0 1.1)
               (0.95 1.1 1.2)
)

%=========================================================================
%  Following matrices are used to implement Project Decision Making Policies
%     including how to determine to whom to report an exception, how to make
%     a decision for an exception, what is the maximum time a participant will
%     wait before it takes delegation by default.
%
%=========================================================================

%-------------------------------------------------------------------
%  Time To Wait For Decision Policy:
%     This matrix defines how long a participant should wait for a decision
%     before it assumes delegation by default. Participants playing different
%     roles in the organization may have different time-out durations.
%
(Matrix TimeToWaitForDecision
    :Row       PM SL ST                        %= Participant roles
    :Values     480                  %= Time-out duration in minutes
             960
             960
)

%-------------------------------------------------------------------
%  Decision Maker Policy:
%     This matrix is used by a participant to determine who should make
%     decision for his/her exception, based on project's centralization
%     policy. The assumption is that more centralized project teams
%     requires higher level participants make decisions for exceptions.
%
(Matrix DecisionMakerPolicy
    :Row       PM SL ST              %= Decision maker's role
    :Column    High Medium Low               %= Level of centralization
    :Values    (0.6 0.2 0.1)         %= Probability
            (0.3 0.6 0.3)              % a certain role should
               (0.1 0.2 0.6)           % make the decision.
)

%-------------------------------------------------------------------
%  Decision Choice Policy:
%     This matrix is used by a decision-maker to determine how an exception should be
dealt with, based on project's centralization policy. NOTE:  The assumption is that
higher level participants (e.g., project managers) tend to make more Rework decisions.
Vité experience has found this assumption reasonable for routine engineering design.
However, for domains like software engineering, Vité staff has found that the reverse is
true.  Participants (hackers) want to fix every known bug, whereas managers want to ship
on time, even with known, non-serious bugs.  This matrix should be adjusted to reflect
the "bug fixing" culture of the organization being modeled.
%
(Matrix DecisionChoicePolicy
    :Row       PM SL ST                        %= Decision-maker's role
    :Column    Rework Quick-Fix Ignore %= Decision type
    :Values    (0.65  0.3 0.05)                %= Probability
            (0.4 0.4 0.2)              % the decision-maker will
               (0.05 0.35 0.6)                 % make a certain type of decision
)

%=========================================================================
%  Information exchange Probability adjustment:
```

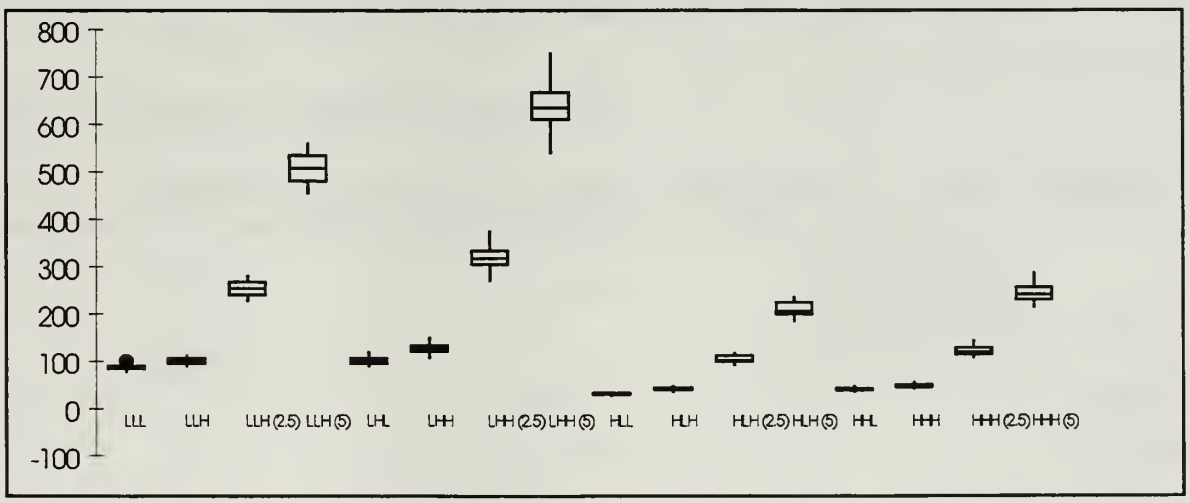
```
%  The following matrices adjust the frequency probability of different types of
information exchange based on the Level of project Formalization:
%
% ?AdjustedInfoXProbability = OriginalCommunicationProbability * AdjustFactor;
%
% The Info Exchange AdjustFactor is retrieved from the following matrix given the level
of formalization.
%
% NOTE: Meeting frequency is not adjustable in Vité-Project, so the Meet row of the
matrices is not meaningful.
%
%=============================================================================

%-----------------------------------------------------------------
%  This matrix defines the VFP adjustment factor for different types of information
exchange.
%
(Matrix CoordinationDistribution
    :Row        InfoX Meet Noise              %= Information exchange type
    :Column     High Medium Low               %= Level of formalization
    :Values     (0.5 1.0 2.0)                 %= Info Exchange AdjustFactor.
                (0.7 1.0 1.0)
                (1.0 1.0 1.0)
)



%=============================================================================
% In Vité-Project, when a participant picks up an information exchange item, it has to
% decide whether to attend the request for information exchange.  This matrix defines the
% chance a participant attends to a given type of information exchange given a level of
% strength of organization matrix.
%
% e.g., if Matrix Strength is High (as in a Project organization), then a participant
% will probabilistically attend to 80% of information exchanges, and 20% of the meetings
% and 20% of the Noise. Project organizations have high Matrix strength; functional teams
% have low matrix strength.
%
%
(Matrix CoordinationPriority
    :Row        InfoX Meet Noise              %= Type of information exchange
    :Column     High Medium Low               %= Org Matrix Strength
    :Values     (0.9 0.7 0.6)                 %= Probability
                (0.6 0.7 0.9)                 % a participant will attend
                (0.2 0.2 0.2)                 % a communication.
)

%=============================================================================
%
% Communications-related matrices
%
%=============================================================================

%-----------------------------------------------------------------
%  This matrix defines the length of time (in minutes) it takes to
%  deliver messages using different communication tools
%

(Matrix ToolTimeToDeliver
    :Row        Conversation Email Fax Memo Phone VideoConf VoiceMail %= Communication
tool
    :Values     10
                1
                1
                5
                1
                1
                1
)
```

```
%-----------------------------------------------------------------------
%  This matrix defines the length of time (in minutes) it takes for
%  messages to expire in the recipients in-tray
%

(Matrix ToolTimeToExpire
    :Row       Conversation Email Fax Memo Phone VideoConf VoiceMail %= Communication
tool
    :Values    60
               2400
               1440
               2400
               5
               10
               960
)

%-----------------------------------------------------------------------
%  This matrix defines the volume (in minutes) for each type of message
%

(Matrix MessageVolume
    :Row       PM SL ST                        %= Recipients role
    :Column    decision exception info_exchange meeting noise %= Message type
    :Values    (10 120 30 0 10)
               (10 240 30 0 10)
               (10 240 30 0 10)
)
%%%%%%%%%%%%%%%
% END OF FILE %
%%%%%%%%%%%%%%%
```

APPENDIX D
STATISTICAL ANALYSIS OF SIMULATION OUTPUTS

A. Descriptive Statistics and Boxplots

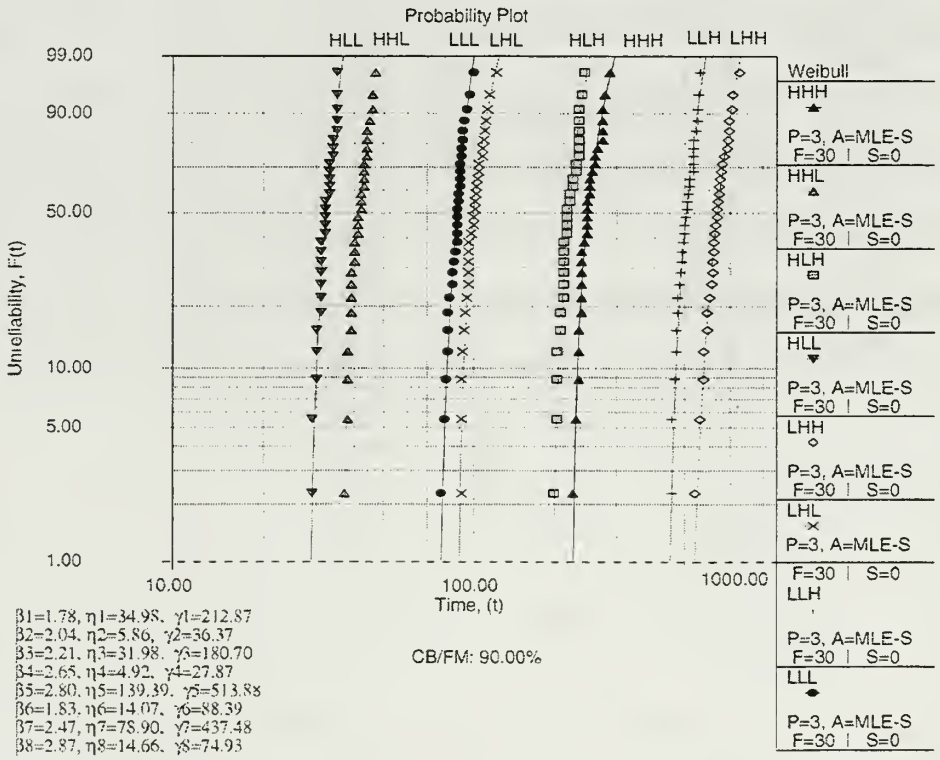
Scenario	Mean	Standard	Median	Mode	Kurtosis	Skewness	Count	Max	Min
LLL	88.00	5.04	88	88	0.104	0.130	30	100	78
LLH	101.47	6.28	101.5	107	-1.164	-0.006	30	112	91
LLH (2.5)	253.93	15.69	254	268	-1.188	0.002	30	280	228
LLH (5)	507.33	31.40	507.5	535	-1.164	-0.006	30	560	455
LHL	100.87	7.25	100.5	96	-0.232	0.562	30	119	91
LHH	127.60	9.79	127	115	-0.172	0.204	30	150	108
LHH (2.5)	319.23	24.51	318	288	-0.183	0.195	30	375	270
LHH (5)	638.00	48.97	635	575	-0.172	0.204	30	750	540
HLL	32.23	1.83	32	31	-0.954	0.063	30	35	29
HLH	41.80	2.77	41	40	-1.134	0.139	30	47	37
HLH (2.5)	104.77	7.02	103	100	-1.153	0.147	30	118	93
HLH (5)	209.00	13.86	205	200	-1.134	0.139	30	235	185
HHL	41.57	2.71	41.5	39	-0.911	0.216	30	47	37
HHH	48.80	3.66	48	46	-0.565	0.600	30	57	43
HHH (2.5)	122.20	9.16	120	115	-0.542	0.617	30	143	108
HHH (5)	244.00	18.31	240	230	-0.565	0.600	30	285	215



The descriptive statistics do not give conclusive information about the kind of distribution observed. The boxplots show that complexity (the third variable) has the strongest influence over the development time, efficiency seems to have less impact, and requirements volatility seems to have moderate influence.

B. Weibull Probability Plots

The data obtained from the simulations was analyzed with a Weibull++ Ver 5.0 from Reliasoft (http://www.reliasoft.com). This statistical analysis tool checks what is the distribution function that better fits the sample. The distributions compared were exponential (one and two parameters), Weibull (two and three parameters), normal, and lognormal. In all the cases the tool found that Weibull with three parameters was the best fit. The tool uses the maximum likelihood method with a confidence of 90% to evaluate which distribution functions had the best fit. The following plot is a Weibull paper and shows the data points as icons and the distribution function recommended by the tool as lines.



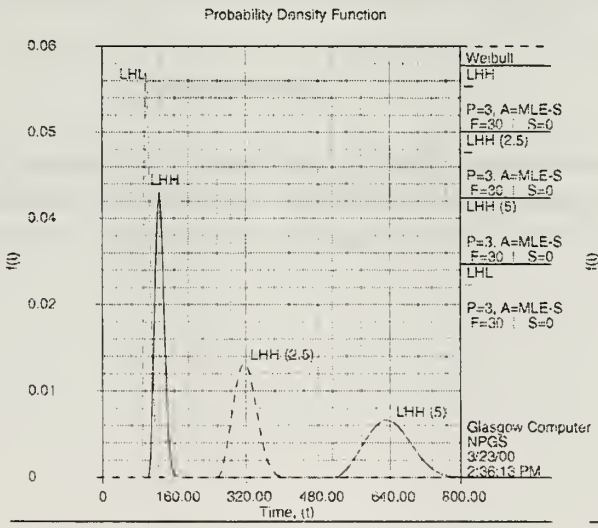
C. Probability Distribution Functions

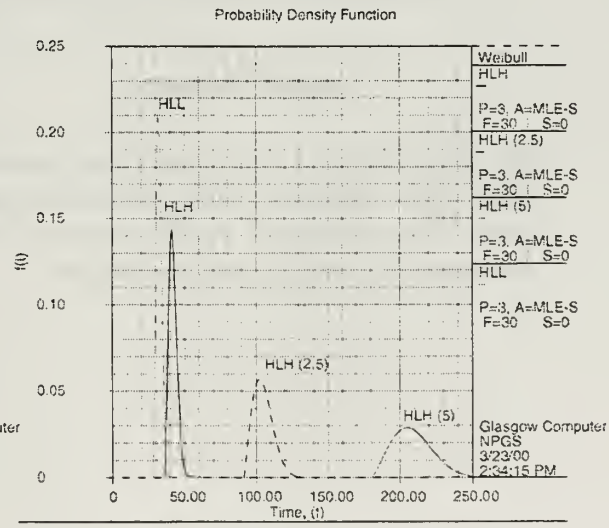
The following table presents the parameters for the Weibull distribution functions that model each scenario. The method used to calculate these parameters was maximum likelihood with 90% of confidence.

	alpha	*beta*	*gamma*
LLL	2.87	14.66	74.93
LLH	2.47	15.78	87.5
LLH 2.5	2.39	38.49	219.87
LLH 5	2.47	78.9	437.48
LHL	1.83	14.07	88.39
LHH	2.8	27.88	102.78
LHH 2.5	2.82	70.26	256.65
LHH 5	2.8	139.39	513.88
HLL	2.65	4.92	27.89
HLH	2.21	6.4	36.14
HLH 2.5	2.09	15.46	91.07
HLH 5	2.21	31.98	180.7
HHL	2.04	5.86	36.37
HHH	1.76	6.94	42.61
HHH 2.5	1.74	17.14	106.93
HHH 5	1.78	34.98	212.87

D. Effect of the Requirements Volatility

The following graphs show the influence of requirements volatility. Two graphs are presented to observe the effect of the variation in requirements volatility.





β1=2.80, η1=27.88, γ1=102.78
β2=2.82, η2=70.26, γ2=256.65
β3=2.80, η3=139.39, γ3=513.88
β4=1.83, η4=14.07, γ4=88.39

β1=2.21, η1=6.40, γ1=36.14
β2=2.09, η2=15.46, γ2=91.07
β3=2.21, η3=31.98, γ3=180.70
β4=2.65, η4=4.92, γ4=27.87

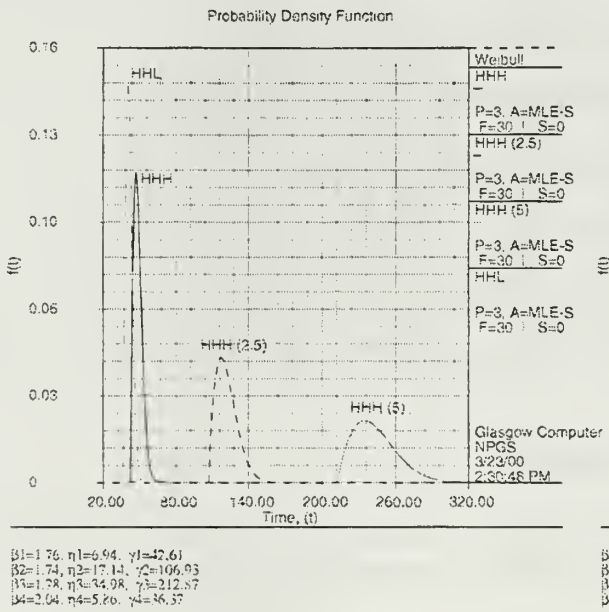
High Requirements Volatility

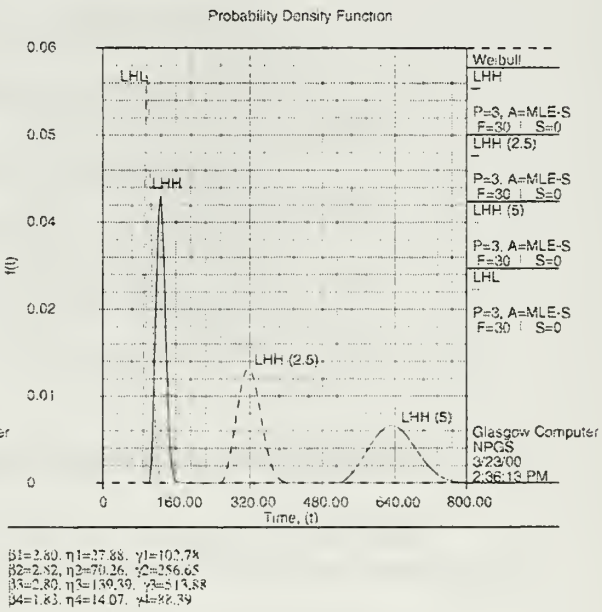
Low Requirement Volatility

In both cases the increment of volatility produces a change in the scale parameter. This shift is magnified when the complexity is high. The effect is also magnified when the efficiency is low.

E. Effect of the Efficiency

The following graphs show the influence of efficiency. Two graphs are presented to discriminate the cases of high and low complexity in order to avoid confounding factors.





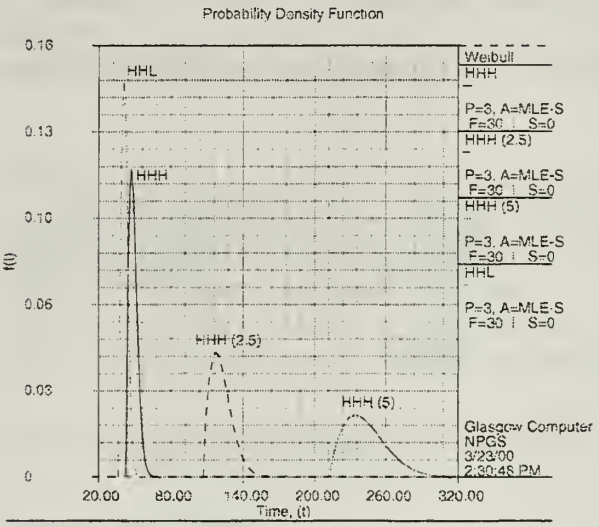
High Efficiency

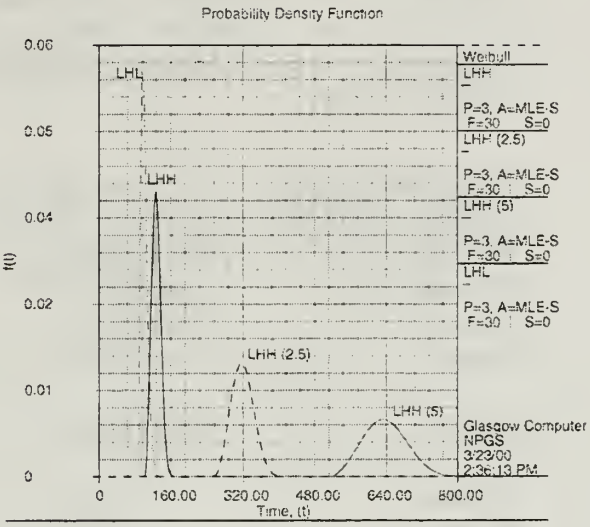
Low Efficiency

For same levels of complexity and requirements volatility, the change in efficiency from high to low modifies mainly the shape parameters, and has also effects over the scale and delay. This effect is more notorious when low efficiency is combined with high requirement volatility and complexity.

F. Effect of the Complexity

The following graphs show the influence of complexity in scenarios with high requirements volatility.





High Efficiency
High Requirement Volatility

Low Efficiency
High Requirement Volatility

In both cases the increment on complexity produces a shift to the right. This shift is magnified when the efficiency is low. For the high efficiency scenarios, one can observe that the shape and scale parameters in the group are relatively stable. The main difference is the delay. The same phenomenon occurs in the four low efficiency scenarios.

G. ANOVA

To analyze the combined effect of the complexity and requirements volatility, we conducted Two-way ANOVA. The results show that the combined effect of complexity and requirements volatility is similar to the random error in the samples. The biggest contributor for the variability is the complexity. The following tables show the Two-way ANOVA results:

ANOVA (EF = H)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample (RV)	17733.20417	1	17733.2	204.9622	9.69E-34	3.881851
Columns (CX)	1382083.513	3	460694.5	5324.754	1.4E-213	2.64351
Interaction	7242.279167	3	2414.093	27.90233	1.92E-15	2.64351
Within	20072.5	232	86.5194			
Total	1427131.496	239				

CX effect	96.84%
RV effect	1.24%
Combined effect	0.51%
Random effect	1.41%

ANOVA (EF = L)

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample (RV)	207035.0042	1	207035	372.7349	3.61E-50	3.881851
Columns (CX)	8810456.713	3	2936819	5287.294	3.2E-213	2.64351
Interaction	125760.5458	3	41920.18	75.47088	4.22E-34	2.64351
Within	128864.0333	232	555.4484			
Total	9272116.296	239				

CX effect	95.02%
RV effect	2.23%
Combined effect	1.36%
Random effect	1.39%

The ANOVA method was also used to test the accuracy of the models. If the hypothesis that the samples obtained from the simulation and the samples obtained from the estimates are from the same population cannot be rejected, then the estimation errors are statistically insignificant. The following tables present the ANOVA for samples from the simulation and samples from the estimation models.

Ho: Both samples belong to the same population

Ha: The samples belong to different populations

$\alpha = 0.05$

Anova: Single Factor

Model 1

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Simulated	16	2984	186.5	30360.8
Estimated Model 1	16	2915.314	182.2071	27657.64

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	***P-value***	*F crit*
Between Groups	147.4293	1	147.4293	0.005082	**0.943641**	4.170886
Within Groups	870276.7	30	29009.22			
Total	870424.1	31				

P-value > α hence Ho cannot be rejected.

Anova: Single Factor

Model 2

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Simulated	16	2984	186.5	30360.8
Estimated Model 2	16	2930.126	183.1329	27929.33

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	***P-value***	*F crit*
Between Groups	90.69892	1	90.69892	0.003112	**0.955883**	4.170886
Within Groups	874351.9	30	29145.06			
Total	874442.6	31				

P-value > α hence Ho cannot be rejected.

Anova: Single Factor

Model 3

SUMMARY

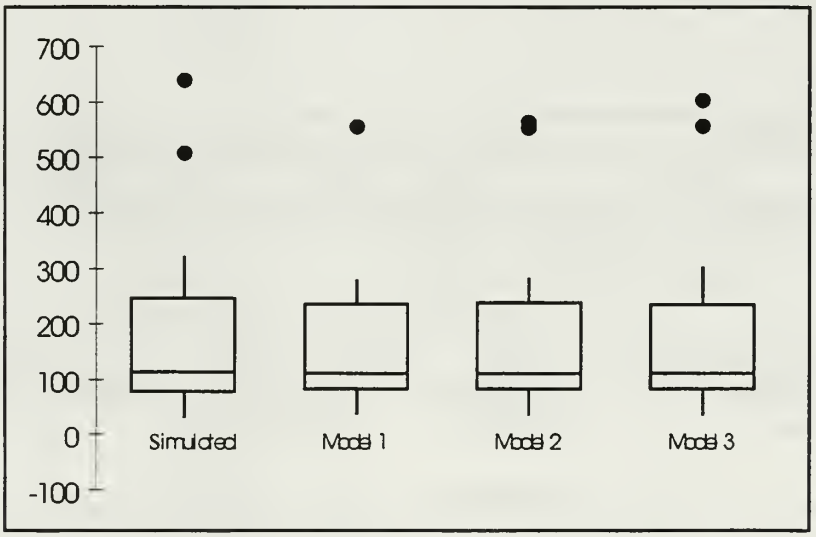
Groups	*Count*	*Sum*	*Average*	*Variance*
Simulated	16	2984	186.5	30360.8
Estimated Model 3	16	2986	186.625	30328.65

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	***P-value***	*F crit*
Between Groups	0.125	1	0.125	4.12E-06	**0.998394**	4.170886
Within Groups	910341.8	30	30344.73			
Total	910341.9	31				

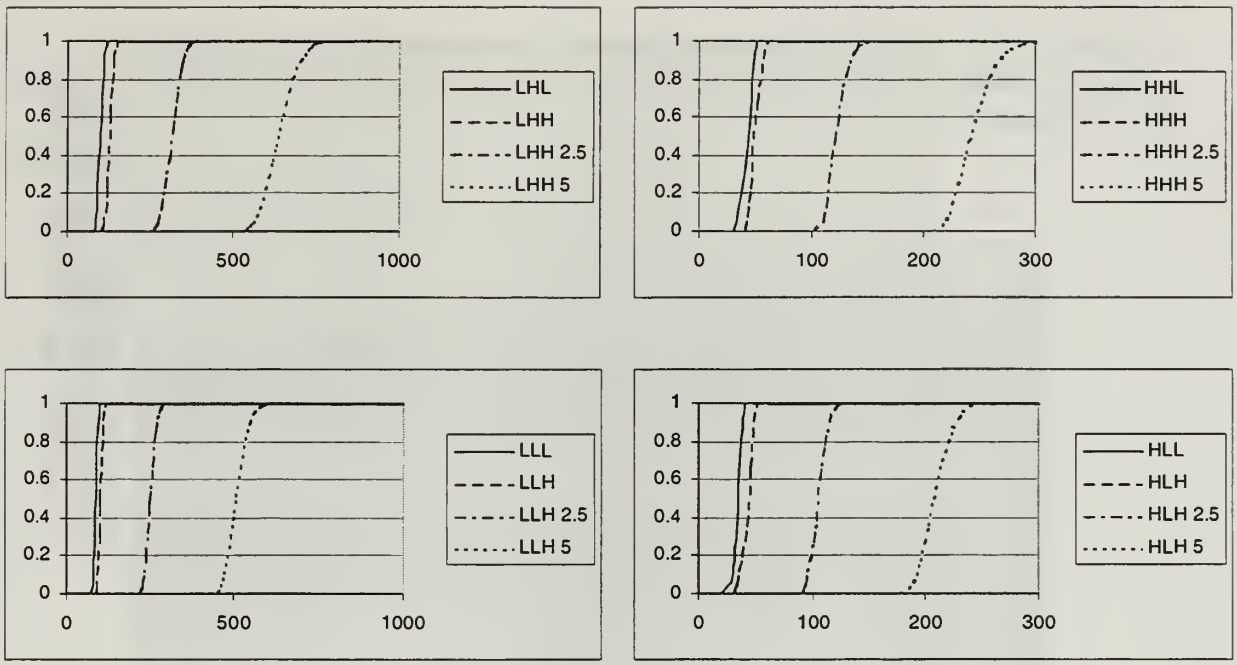
P-value > α hence Ho cannot be rejected.

The following boxplot show the increasing accuracy in the three models.



H. Cumulative Density Functions (CDF) and Stochastic Dominance

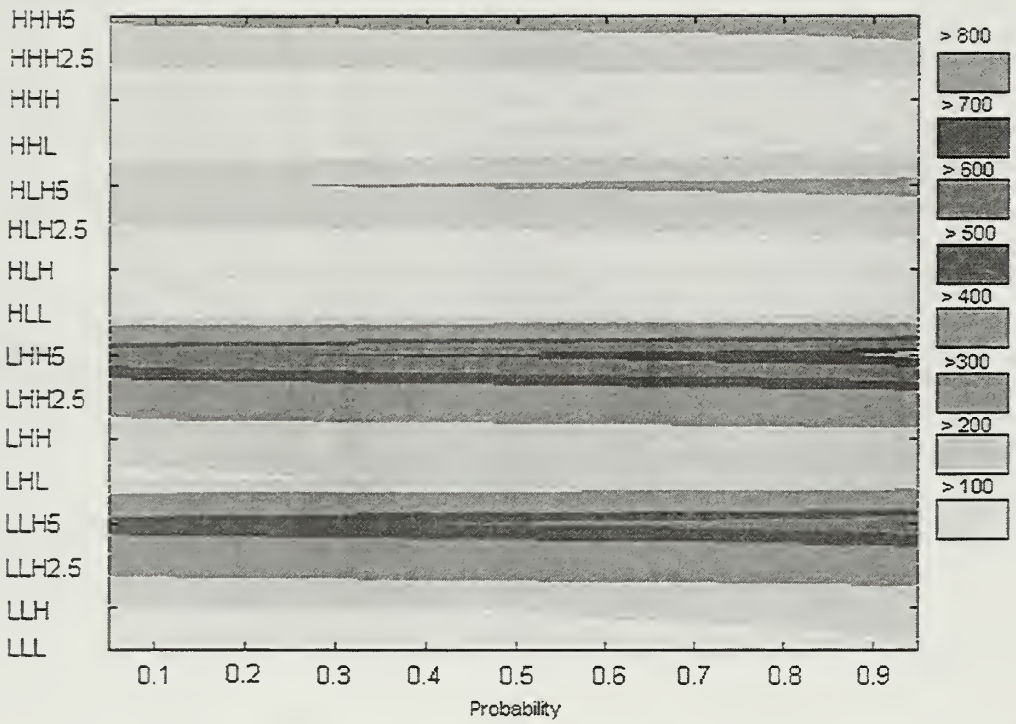
The analysis of stochastic dominance (see Appendix F) reveals that under similar complexity and requirements volatility, high efficiency scenarios are always dominant. Under similar levels of complexity and efficiency, then the dominance is determined by the less volatility in the requirements. Finally, under similar conditions of efficiency and requirements volatility, the stochastic dominance depends on the complexity. By using stochastic dominance the decision-maker can decide between alternatives. First the decision-maker should apply the estimation model for each alternative to obtain a set of parameters α, β, and γ for each alternative. Then each alternative can be compared for stochastic dominance using the Weibull cdfs.



I. Contour of Time

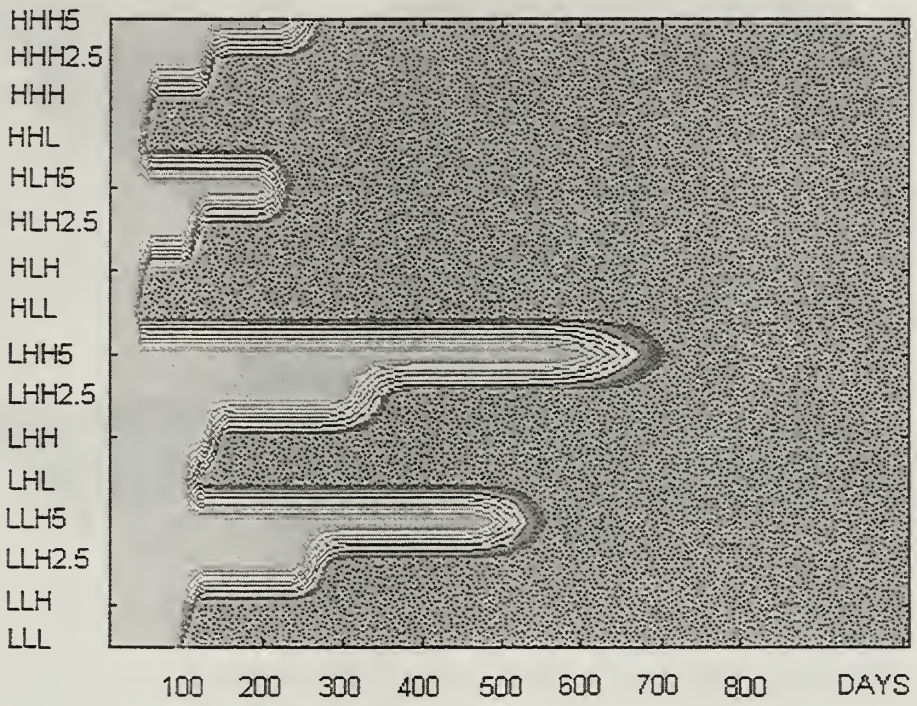
One of the difficulties in visualizing the model is that it has four variables (efficiency, requirements volatility, complexity, and time), hence it is necessary a five dimensional space to represent it (four dimensions for the parameters plus one extra dimension for the scalar value of the probability associated).

The following graph represents the lines of same expected time given a discrete set of scenarios with different efficiency, complexity, and requirements volatility. The graph is only useful to visualize the combined effect of the three parameters of the model. Given a certain scenario and a confidence probability it is possible to determine the expected time in days. For instance, the comparison of HHH5 (high efficiency, high volatility, high complexity) vice LHH5 (low efficiency and the same other parameters) show the effect of efficiency.


HHH5
HHH2.5
HHH
HHL
HLH5
HLH2.5
HLH
HLL
LHH5
LHH2.5
LHH
LHL
LLH5
LLH2.5
LLH
LLL
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Probability
> 800
> 700
> 600
> 500
> 400
>300
> 200
> 100

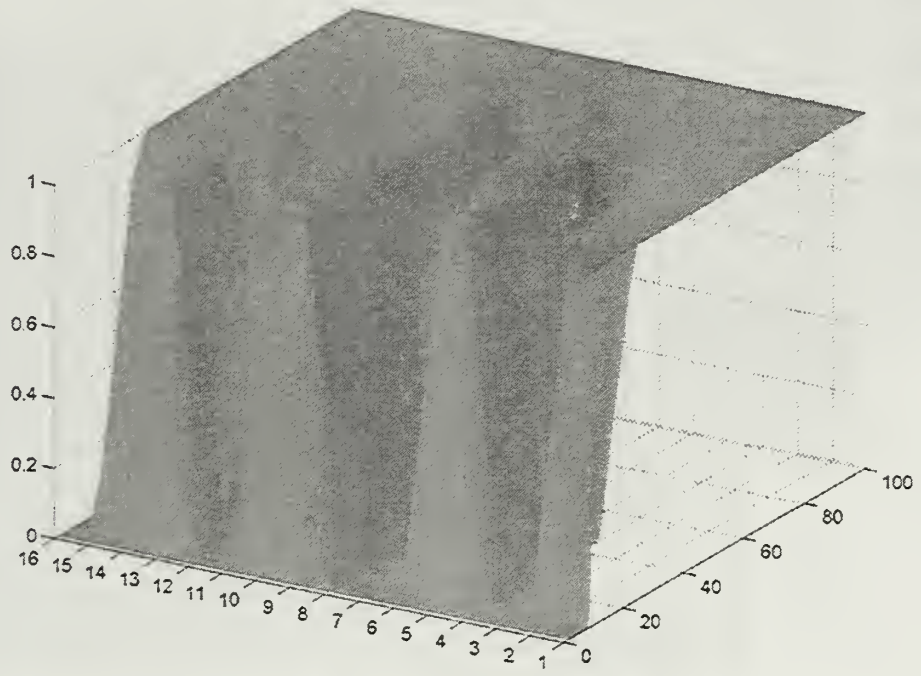
J. Contour of Probabilities

The following graph represents the lines of same probability of finishing the project at a given date, given a discrete set of scenarios with different efficiency, complexity, and requirements volatility. The graph is only useful to visualize the combined effect of the three parameters of the model.



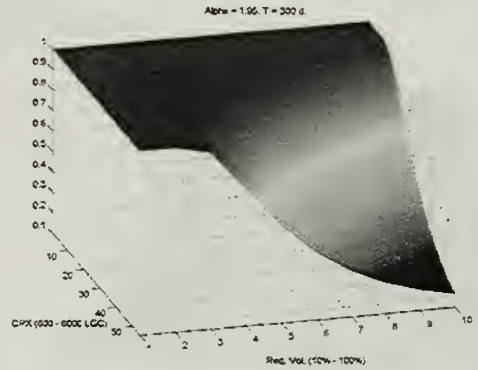
K. Surface of Cumulative Distribution

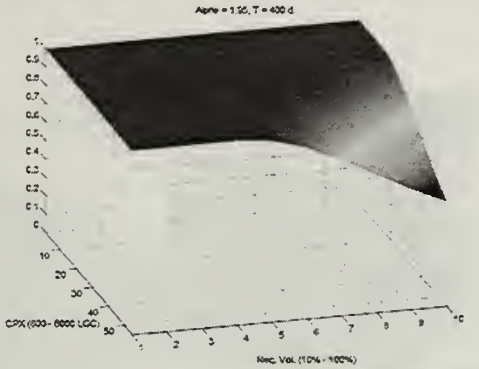
The following graph represents a 3D view of the cumulative distributions for a discrete set of scenarios. The z-axis represents the cdf, the x-axis represents the scenario (1-16), and the y-axis represents the time (0-100).

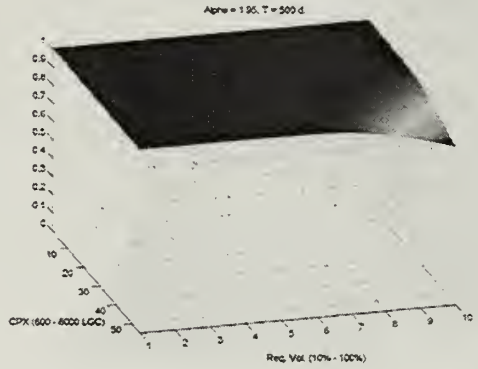


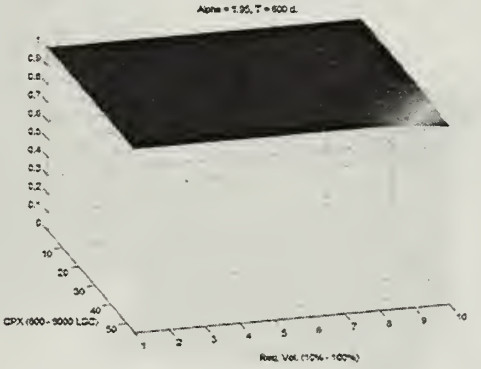
L. Snapshots of the Surface of Cumulative Distribution for High Efficiency

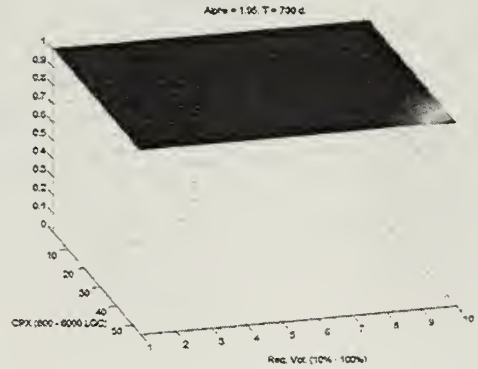
The following series of graphs represents the continuous 3D aspect of the five-dimension model given a high efficiency scenario for five different moments in time. The axes represent complexity, volatility, and cdf. The five snapshots represent time in a discrete way. Efficiency is constant and high for all the graphs.





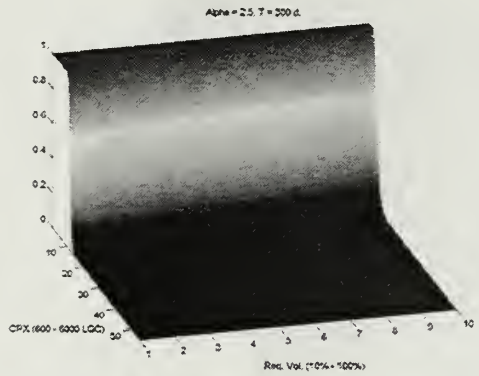


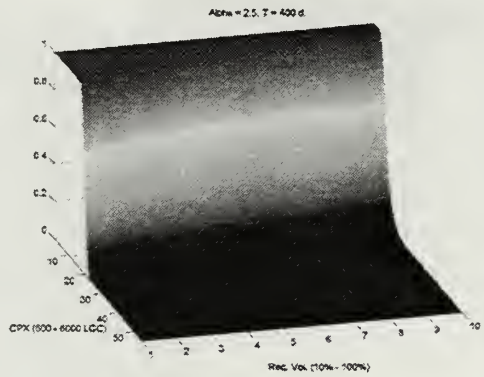


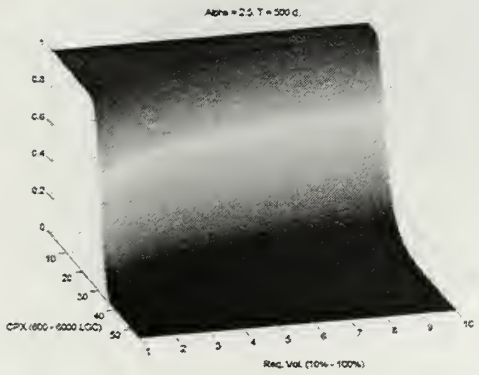


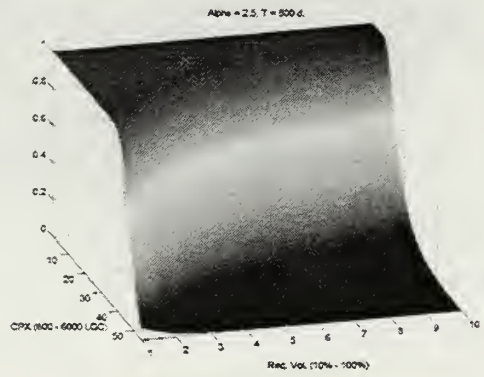
M. Snapshots of the Surface of Cumulative Distribution for High Efficiency

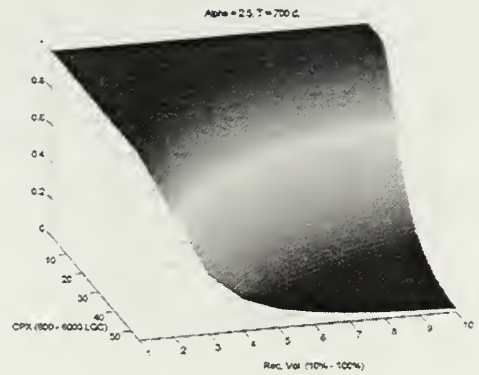
The following series of graphs represents the continuous 3D aspect of the five-dimension model given a high efficiency scenario for five different moments in time. The axes represent complexity, volatility, and cdf. The five snapshots represent time in a discrete way. Efficiency is constant and low for all the graphs.





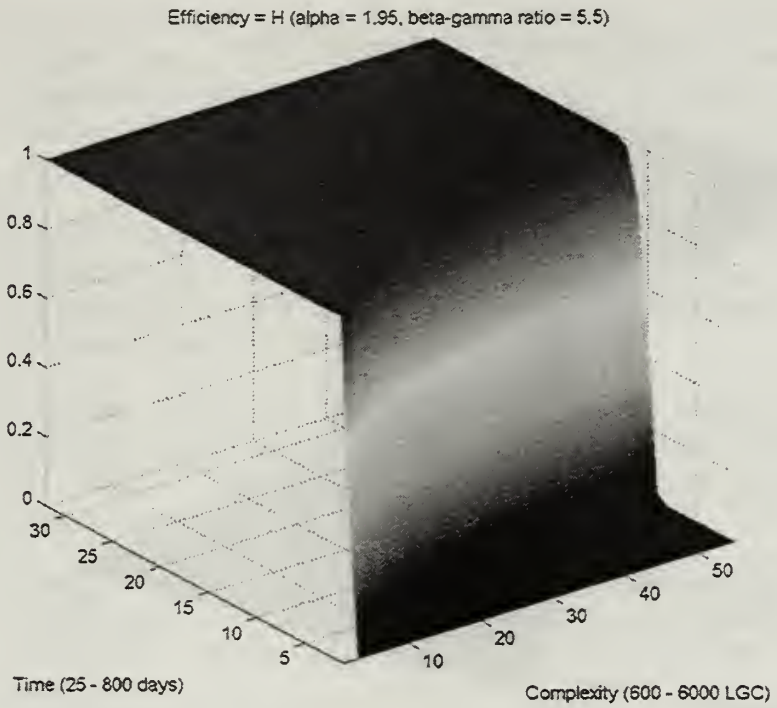






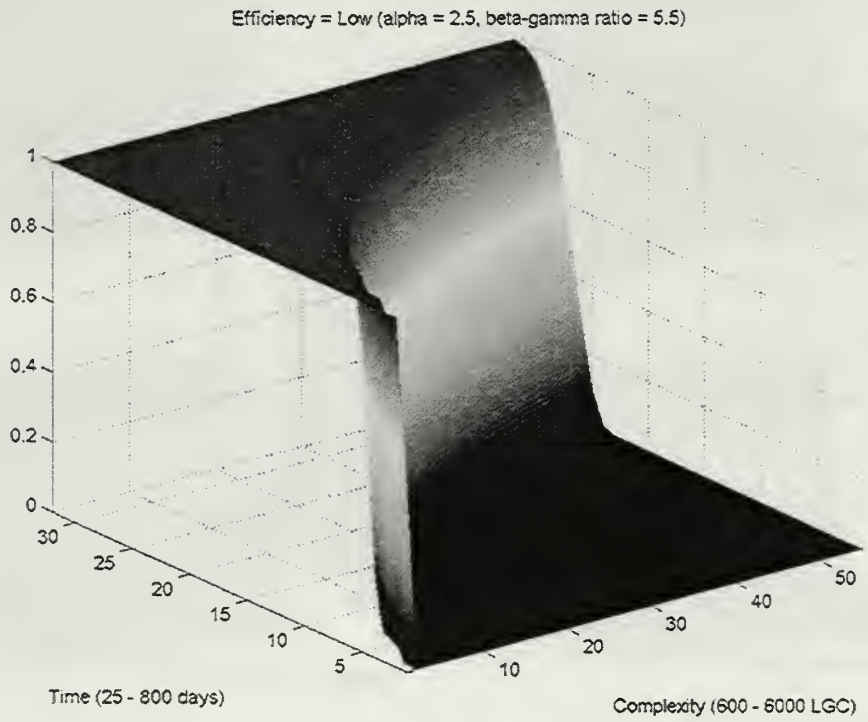
N. Surface of Cumulative Distribution for High Efficiency and Gamma-Beta ratio = 5.5

The following graph represents the cdf surface for a given level of efficiency and a given level of volatility. The three axes correspond to complexity, time, and cdf. This graph predicts the future of the project under the hypothesis of constant volatility and high efficiency.



O. Surface of Cumulative Distribution for Low Efficiency and Gamma-Beta Ratio = 5.5

The following graph represents the cdf surface for a given level of efficiency and a given level of volatility. The three axes correspond to complexity, time, and cdf. This graph predicts the future of the project under the hypothesis of constant volatility and low efficiency.



APPENDIX E

REFERENCES ABOUT VITEPROJECT, VDT AND THEIR VALIDATION

1. Dym, Clive, and Levitt, Raymond E., Knowledge-based Systems in Engineering, McGraw-Hill Book Co., 1991.

2. Levitt, R.E., G.P. Cohen, J.C. Kunz, C.I. Nass, T. Christiansen, and Y. Jin, "The Virtual Design Team: SImulating How Organization Structure and Information Processing Tools Affect Team Performance," in Carley, K.M. and M.J. Prietula, editors, Computational Organization Theory, Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1994.

3. Levitt, Raymond E., Yan Jin, Gaye Oralkan, John Kunz and Tore Christiansen, "Computational Enterprise Models: Toward Analysis Tools for Designing Organizations," in Coordination Theory and Collaboration Technology, Gary Olson, Thomas Malone eds., 1996.

4. Logcher, Robert D., and Raymond E. Levitt, "Organization and Control of Engineering Design Firms," ASCE Engineering Issues, Vol. 105, No. EI1, pp. 7-14, January 1979, pp. 7-14 (reprinted in Italian and English in L'Industria delle Costruzioni, Rome, Italy: March 1982).

5. Levitt, Raymond E., "Superprojects and Superheadaches: Balancing Technical Economies of Scale Against Management Diseconomies of Size and Complexity," Project Management Journal, Vol. 15, No. 4, December 1984, pp. 82-89.

6. Levitt, Raymond E., and John C. Kunz, "Using Knowledge of Construction and Project Management for Automated Schedule Updating," Project Management Journal, Vol. XVI, No. 5, December 1985, pp. 57-76.

7. Levitt, Raymond E., and John C. Kunz, "Using Artificial Intelligence Techniques to Support Project Management," Artificial Intelligence in Engineering Design, Analysis and Manufacturing, 1 (1), 1987, pp. 3-24.

8. Kartam, Nabil, and Raymond E. Levitt, "An Artificial Intelligence Approach to Project Planning under Uncertainty," Project Management Journal, Vol. 22, No. 2, June 1991, pp. 7-11.

9. Kartam, N. A., R. E. Levitt and D. E. Wilkins, "Extending Artificial Intelligence Techniques for Hierarchical Planning," ASCE Journal of Computing in Civil Engineering, Vol. 5, No. 4, October 1991, pp. 464-477.

10. Dym, C. L., and R. E. Levitt, "Toward the Integration of Knowledge for Engineering Modeling and Computation," Engineering with Computers, Vol. 7, No. 4, pp. 209-224, Fall 1991.

11. Levitt, Raymond E., Yan Jin and Clive L. Dym, "Knowledge-based Support for Management of Concurrent, Multidisciplinary Design," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 5, No. 2, 1991, pp. 77-95.

12. Winstanley, Graham, Michael A. Chacon, and Raymond E. Levitt, "An integrated project planning environment," Intelligent Systems Engineering, Vol. 2, No. 2, 1993, pp.91-106.

13. Jin, Yan and Raymond E. Levitt, "i-AGENTS: Modeling Organizational Problem Solving in Multi-Agent Teams," Intelligent Systems in Accounting, Finance and Management, Vol. 2, 1993, pp. 247-270.

14. Jin, Yan, Raymond E. Levitt, Tore Christiansen and John C. Kunz, "The Virtual Design Team: Modeling Organizational Behavior of Concurrent Design Teams," Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 9, No.2, April, 1995, pp. 145-158.

15. Levitt, Raymond E., and Robert D. Logcher, "The Human Element in Project Control Systems," Project Management Institute Annual Symposium, Montreal, Canada, October 1976, pp 233-237.

16. Logcher, Robert D., and Raymond E. Levitt, "Human Information-Handling Capacity as a Factor in the Design of Project Control Systems," USNCCIB International Symposium on Organization and Management of Construction, Vol. II, Washington, DC, May 1976, pp. 136-146.

17. Levitt, Raymond E., and Robert D. Logcher, "Principles for an Integrated MIS for Design Firms," Second International Symposium on Organization and Management of Construction of CIB-W65, Vol. III, Haifa, Israel, October 31 - November 2, 1978, pp. 225-238.

18. Levitt, Raymond E., "Superprojects and Superheadaches: Balancing Technical Economies of Scale Against Management Diseconomies of Size and Complexity," Project Management Institute/INTERNET Joint Annual Symposium, Boston, MA, September 1981, pp. 487-491.

19. Levitt, Raymond E., and John C. Kunz, "A Knowledge-Based System for Updating Engineering Project Schedules," WAM/ASME symposium on applications of knowledge-based systems to engineering analysis and design, Miami Beach, FL, published in Applications of Knowledge-Based Systems to Engineering Analysis and Design, C. L. Dym, Editor, ASME Proceedings Vol. AD-10, November 1985, pp. 47-65.

20. Kunz, John C., Thomas Bonura, Raymond E. Levitt and Marilyn J. Stelzner, "Contingent Analysis for Project Management Using Multiple Worlds,"

Proceedings of the First International Conference on Applications of Artificial Intelligence to Engineering Problems, Southampton, UK: April 15-18, 1986, pp. 707-718.

21. Levitt, Raymond E., John C. Kunz and Nabil A. Kartam, "Using Artificial Intelligence Techniques for Automated Planning and Scheduling," Proceedings of the Fourth International Symposium on Robotics and Artificial Intelligence in Building Construction, Building Research Station, Technion IIT, Haifa, Israel, June 1987, pp. 27-52.

22. Levitt, Raymond E., "AI in Project Management - Initial Attempts," in Artificial Intelligence in the Engineering and Management of Major Projects: Hope or Reality?, proceedings of workshop sponsored by Major Projects Association, London, September 1987, pp. 45-60.

23. Kartam, Nabil A., and Raymond E. Levitt, "An Analysis of Artificial Intelligence Planning Systems," in Hamza, M. H., Ed., Expert Systems Theory and Applications, Proceedings of The International Association of Science and Technology for Development (IASTED), December 12-14, 1988, Los Angeles, CA, pp. 130-133.

24. Kartam, N. A., R. E. Levitt and D. E. Wilkins, "A Centralized Approach for Representing and Resolving Interactions among Multi-agent Tasks while Planning Hierarchically," IEEE, March 1990, pp. 250-256.

25. Cohen, G. P., and R. E. Levitt, "The Virtual Design Team: An Object-oriented Model of Information-sharing in Project Design Teams," ASCE Construction Congress, Expert Systems Symposium in Computer-integrated Design and Construction, Cambridge, Massachusetts, April 1991.

26. Jin, Y., J.C. Kunz, R.E. Levitt and G. Winstanley, "Design of Project Plans from Fundamental Knowledge of Engineering Systems," Proceedings of the AAAI Fall Symposium-Design from Physical Principles, Cambridge, MA, October 1992, pp. 149-154.

27. Kunz, J.C., G.P. Cohen, and R.E. Levitt, "Using an Organizational Model to Predict Effects on Design Team Productivity," AAAI 93 Workshop on Design, August, 1993.

28. Kunz, J.C., G.P. Cohen, and R.E. Levitt, "Modeling Effects of Organizational Structure and Communication Tools on Design Team Productivity," AAAI 93 Workshop on Organizational Modeling, August, 1993.

29. Jin, Yan, Raymond E. Levitt, and Tore Christiansen, "The Virtual Design Team: A Computer Simulation Framework for Studying Organizational Aspects of Concurrent Design," 1994 International Conference on Simulation in Engineering Education, January 23-27, 1994, Tempe, Arizona.

30. Jin, Yan, Raymond E. Levitt, Tore Christiansen, and John C. Kunz, "The Virtual Design Team: A Computational Model of Engineering Design Teams," AAAIâ94 Spring Symposium on Computational Organization Design, March, 1994, Stanford, California.

31. Levitt, Raymond E., "Problems in the Organization of Very Large Projects," Gruppo di Ricerca sul Management (G.R.M.), Quaderni #21, Rome, April 1985.

32. Levitt, Raymond E., Clive L. Dym and Yan Jin, "Knowledge-Based Support for Concurrent, Multidisciplinary Design," CIFE Working Paper #10, Center for Integrated Facility Engineering, Department of Civil Engineering, Stanford University, Stanford, CA, January 1991.

33. Levitt, Raymond E., G.P. Cohen, J.C. Kunz, C.I. Nass, "The "Virtual Design Team": Using Computers to Model Information Processing and Communication in Organizations," CIFE Working Paper #16, Center for Integrated Facility Engineering, Department of Civil Engineering, Stanford University, Stanford, CA, July 1992.

34. Jin, Yan, Raymond E. Levitt, and Tore Christiansen, "The Virtual Design Team: A Computer Simulation Framework for Studying Organizational Aspects of Concurrent Design," CIFE Working Paper #24, Center for Integrated Facility Engineering, Department of Civil Engineering, Stanford University, Stanford, CA, November 1993 [1993.RO32] Levitt, Raymond E., Geoffrey P. Cohen, John C. Kunz, Clifford I. Nass, Tore Christiansen, and Yan Jin, "The Virtual Design Team: Simulating How Organization Structure and Information Processing Tools Affect Team Performance," CIFE Technical Report #83, Center for Integrated Facility Engineering, Stanford University, Stanford, California, April 1993.

35. Oralkan, Gaye A., Yan Jin, and Raymond E. Levitt, "Modeling Organizational Change in Response to Information Technology," CIFE Working Paper #26, Center for Integrated Facility Engineering, Stanford University, Stanford, California, February 1994.

36. Levitt, Raymond E., Tore R. Christiansen, Geoff P. Cohen, Yan Jin, John C. Kunz, Clifford I. Nass, "The Virtual Design Team: A Computational Simulation Model of Project Organizations," CIFE Working Paper #29, Center for Integrated Facility Engineering, Stanford Univeristy, Stanford, California, March 1994.

37. Levitt, Raymond E., Yan Jin, Gaye A. Oralkan, John C. Kunz, Tore R. Christiansen, "Computational Enterprise Models: Towards Analysis Tools for Designing Organizations," CIFE Working Paper #36, Center for Integrated Facility Engineering, Stanford University, Stanford, California, (37 pages) February, 1995.

38. Levitt, R.E., G.P. Cohen, J.C. Kunz, C.I. Nass, T. Christiansen, and Y. Jin, "The 'Virtual Design Team': Simulating How Organization Structure and Information Processing Tools Affect Team Performance," in Carley, K.M. and M.J. Prietula,

editors, Computational Organization Theory, Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1994.

39. Levitt, Raymond E., Tore C. Christiansen, Geoff P. Cohen, Yan Jin, John C. Kunz, Clifford I. Nass, Jan Thomsen, "The Virtual Design Team: A Computational Simulation Model of Project Organizations", in Kunz, John C., and Raymond E. Levitt, Editors, Integrated Facility Engineering: Research from the First Ten Years of the Center for Integrated Facility Engineering, Gordon Breach, Publishers, Palo Alto, CA, 1999, pp. 253 292.

40. Jin, Yan and Raymond E. Levitt, "i-AGENTS: Modeling Organizational Problem Solving in Multi-Agent Teams," Intelligent Systems in Accounting, Finance and Management, Vol. 2, 1993, pp. 247-270.

41. Jin, Yan, Raymond E. Levitt, Tore Christiansen, and John C. Kunz, "The Virtual Design Team: A Computer Simulation Framework for Studying Organizational Aspects of Concurrent Design," Simulation, Vol.64, No.3, pp.160-174, March, 1995.

42. Jin, Yan, Raymond E. Levitt, Tore Christiansen and John C. Kunz, "The Virtual Design Team: Modeling Organizational Behavior of Concurrent Design Teams," Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 9, No.2, April, 1995, pp. 145-158.

43. Jin, Yan, and Raymond E. Levitt, "The Virtual Design Team: A Computational Model of Project Organizations,"Journal of Computational and Mathematical Organization Theory 2 (3), Fall, 1996, pp. 171-195.

44. Kunz, J. C., T. R. Christiansen, G. P. Cohen, Y. Jin, and R. E. Levitt, "The Virtual Design Team: A Computational Simulation Model of Project Organizations," Communications of the Association for Computing Machinery, 41 (11), 1998, pp. 84-92.

45. Levitt, Raymond E., "Toward Analysis Tools for the Engineering Process," Journal of Artificial Intelligence in Engineering Design, Analysis and Manufacturing, Vol. 12, pp.77-88, 1998.

46. Kunz, John C., Tore R. Christiansen, Geoff P. Cohen, Yan Jin, Raymond E. Levitt, "The Virtual Design Team: A Computational Simulation Model of Project Organizations," Communications of the Association for Computing Machinery (CACM) 41 (11), November, 1998, pp. 84-91.

47. Nasrallah, Walid, Peter Glynn and Raymond E. Levitt, "Diversity and Popularity in Organizations and Communities," Journal of Computational and Mathematical Organization Theory, 4 (4), 1998.

48. Christensen, L., Christiansen, T.R., Jin, Y., Kunz, J.C. & Levitt, R.E., "Modelling and Simulating Coordination in Projects," IEEE Journal of Organizational Computing, 9.(1), 1999, pp.33-56.

49. Cohen, G. P., and R. E. Levitt, "The Virtual Design Team: An Object-oriented Model of Information-sharing in Project Design Teams," ASCE Construction Congress, Expert Systems Symposium in Computer-integrated Design and Construction, Cambridge, Massachusetts, April 1991.

50. Jin, Y., J.C. Kunz, R.E. Levitt and G. Winstanley, "Design of Project Plans from Fundamental Knowledge of Engineering Systems," Proceedings of the AAAI Fall Symposium-Design from Physical Principles, Cambridge, MA, October 1992, pp. 149-154.

51. Jin, Y., and R.E. Levitt, "Model-Based and Interactive Planning for Predictive Plant Maintenance Management," Fifth International Conference on Computing in Civil Engineering, Anaheim, CA, June 1993, pp 1830-1837.

52. Kunz, J.C., G.P. Cohen, and R.E. Levitt, "Using an Organizational Model to Predict Effects on Design Team Productivity," AAAI 93 Workshop on Design, August, 1993.

53. Kunz, J.C., G.P. Cohen, and R.E. Levitt, "Modeling Effects of Organizational Structure and Communication Tools on Design Team Productivity," AAAI 93 Workshop on Organizational Modeling, August, 1993.

54. Jin, Yan, Raymond E. Levitt, and Tore Christiansen, "The Virtual Design Team: A Computer Simulation Framework for Studying Organizational Aspects of Concurrent Design," 1994 International Conference on Simulation in Engineering Education, January 23-27, 1994, Tempe, Arizona.

55. Jin, Yan, Raymond E. Levitt, Tore Christiansen, and John C. Kunz, "The Virtual Design Team: A Computational Model of Engineering Design Teams," AAAI'94 Spring Symposium on Computational Organization Design, March, 1994, Stanford, California.

56. Levitt, Raymond E., "The Virtual Design Team, Modeling High Velocity Project Teams in Construction and Product Development," First International Conference on Computational and Mathematical Organization Theory, Monterrey, Mexico, Oct 30, 1996.

57. Levitt, Raymond E., "Organizational Analysis and Design Tools: State of the Art," First International Conference on Computational and Mathematical Organization Theory, Monterrey, Mexico, Oct 30, 1996.

58. Levitt, Raymond E., "The Virtual Design Team: Designing Organizations and Work Processes like we Design Bridges and Chips," IEEE Workshop on Electronic Design Processes, April 23-25, 1997.

59. Levitt, Raymond E., "Toward Analysis Tools for the Engineering Process," Conference on Computing Futures in Engineering Design, Claremont, CA, May 2-3, 1997, pp. 11-14.

60. Thomsen, J., J. Kunz, Y. Kwon, S. Miller, R. Levitt, "Designing Quality into Product Development Organizations Through Computational Organizational Modeling and Simulation," Workshop on Computational and Mathematical Organization Theory, INFORMS, San Diego CA, May 3, 1997, pp12-14.

61. Thomsen, J., Kwon, Y., Kunz, J. C., and Levitt, R. E., "Simulating the Effects of Goal Incongruency on Project Team Performance." Fourth Congress on Computing in Civil Engineering, ASCE, June 17-19, 1997, Washington DC, pp. 643-650.

62. Levitt, R.E., J. Thomsen and Y. K. Kwon, "Promoting or Discouraging Conflict in Engineering Project Teams, Insights from a Computational Organization Design Perspective," Workshop on Computational and Mathematical Organization Theory, INFORMS, San Diego CA, May 3, 1997, pp. 26-28.

63. CIFE WP 047: "The Virtual Design Team: A Proposed Trajectory of Validation Experiments for Computational Emulation Models of Organizations" Jan Thomsen, Raymond E. Levitt, John C. Kunz, and Clifford I. Nass.

64. CIFE WP 046: "The Virtual Design Team: Designing Quality into Project Organizations through Computational Organizational Simulation" Jan Thomsen, John C. Kunz, and Raymond E. Levitt.

65. CIFE WP 045: "The Virtual Design Team: The Virtual Team Alliance (VTA): Extending Galbraith's Information-Processing Model to Account for Goal Incongruency" Jan Thomsen, Raymond E. Levitt, and Clifford I. Nass.

66. CIFE WP 044: "The Virtual Design Team: The Virtual Team Alliance (VTA): An Extended Theory of Coordination in Concurrent Product Development Projects" Jan Thomsen, Martin A. Fischer, and Raymond E. Levitt.

67. CIFE WP 043: "The Virtual Design Team: A Computational Model of Project Organizations" Yan Jin, Raymond Levitt.

68. CIFE Working Paper 36: "Computational Enterprise Models: Towards Analysis Tools for Designing Organizations" Raymond E. Levitt, Yan Jin, Gaye A. Oralkan, John C. Kunz, Tore R. Christiansen (February, 1995).

69. CIFE Working Paper 29: "The Virtual Design Team: A Computational Simulation Model of Project Organizations" Raymond E. Levitt, Tore Christiansen, Geoff Cohen, Yan Jin, John Kunz, Clifford Nass (March, 1994).

70. CIFE Working Paper 27: "Modeling Organizational Problem Solving in Multiagent Teams" Yan Jin, Raymond Levitt (February, 1994).

71. CIFE Working Paper 26: "Modeling Organizational Change in Response to Informational Technology" Gaye Oralkan, Yan Jin, Raymond Levitt (February, 1994).

72. CIFE Working Paper 24: "The Virtual Design Team: A Computer Simulation Framework for Studying Organizational Aspects of Concurrent Design". Yan Jin, Raymond Levitt, Tore Christiansen (November, 1993).

73. CIFE Working Paper 15: "The Virtual Design Team - Using Simulation of Information Processing to Predict the Performance of Project Teams" Tore Christiansen, Raymond Levitt (July 1992).

74. CIFE Working Paper 16: "The Virtual Design Team: Using Computers to Model Information Processing and Communication in Organizations" Raymond E. Levitt, Geoffrey P. Cohen, John C. Kunz, Clifford I. Nass (July 1992).

APPENDIX F
STOCHASTIC DOMINANCE

The major areas of application of dominance have been finance, insurance, and economics. The classical portfolio problem was the catalyst for the initial research. From there the technique was applied to other domains (Whitmore & Findley, 1978). Stochastic dominance is a methodology related to decision theory. It is based on formal concepts and theorems and employs partial information on the decision-maker's preferences and the random variables to produce a partial ordering (Levy, 1998). The concept of stochastic dominance is used in Appendix D to analyze the effects of the three parameters of our model by comparing the CDFs of different scenarios.

Definition of dominance: Let D be a domain constituted by a set of decisions. Let x be a random variable representing the outcome for a specific decision. Let $d \in D$. We say that the decision d_i dominates the domain D (expressed as d_i DOM D), if and only if the return value for the application of d is maximum for all possible values of x and for all possible $dj \in D$.

$$(\forall d \in D)(\forall x \in X)(R(x, d_i) \geq R(x, d_j)) \Leftrightarrow d_i \text{ DOM } D$$

where D = set of alternatives or decisions, also called Feasible Set (F.S.)

X = set of possible values for the random value x.

R(x, d) = a function that measures the outcome of the decision.

Definition of Efficient Set (E.S.): E.S. is the set of dominating decisions.

$(\forall d \in D)\ (d \text{ DOM } D) \Leftrightarrow d \in \text{E.S.}$

Definition of Inefficient Set (I.S.): I.S. is the set of dominated decisions.

$\text{F.S.} = \text{E.S.} \cup \text{I.S.}$

Definition of First Degree Stochastic Dominance (FSD): FSD is the dominance that can be established by the application of the following definition:

Let F(x) and G(x) be cumulative distribution functions (cdf) related to the decisions f and g respectively. We say that f dominates FSD g (f FSD g) if and only if the $F(x) \geq G(x)$ for all values of x.

$(\forall x)\ (F(x) \geq G(x)) \Leftrightarrow (f \text{ FSD } g)$

Observations:

(1) FSD requires that distributions do not intercept, but can be tangent.

(2) When more than two alternatives exist, the mere condition of being dominated by one alternative is sufficient condition to belong to I.S.

(3) All alternatives in E.S. must intercept, and should not be dominated.

Figure G.1 shows an example of inexistence of FSD. cdf1 and cdf2 belong to E.S. cdf3 is clearly dominated so it belongs to I.S. Note that neither cdf1 or cdf2 dominates each other.

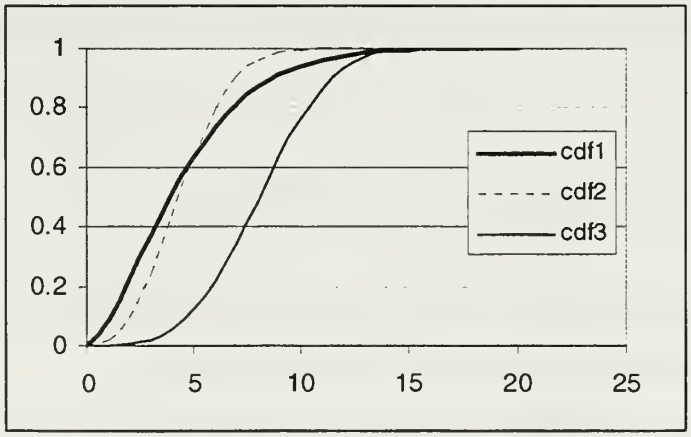


Figure F.1: Concept of Domination. cdf1 dominates cdf3

Definition of sufficient conditions for FSD: Let f, g be two alternatives related to F(x) and G(x) respectively.

(1) We say that f dominates first degree stochastic g (f FSD g), if the maximum range of F(x) is less or equal the minimum range of G(x) (Fig. F.2).

$\text{Max}(\text{Range}(F(x))) \leq \text{Min}(\text{Range}(G(x))) \Leftrightarrow (f \text{ FSD } g)$

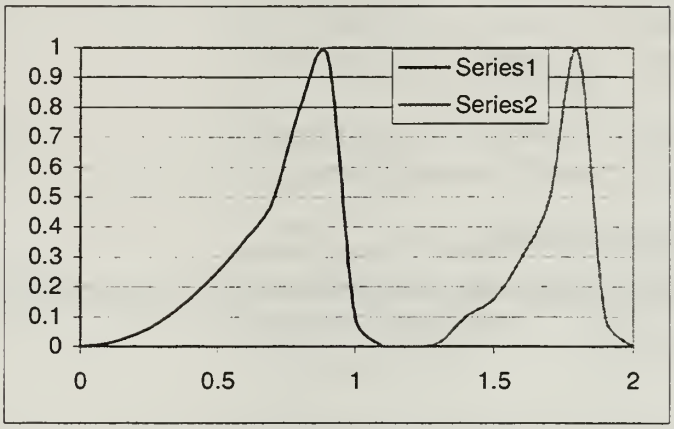


Figure F.2: FSD Sufficient Condition. Series1 dominates Series2

(2) We say f dominates first degree stochastic g (f FSD g), if for all values of x F(x) is greater or equal to G(x) (Fig. F.3).

$(\forall x \in X)(F(x) \geq G(x)) \wedge (\exists y \in X)(F(y) > G(y))$

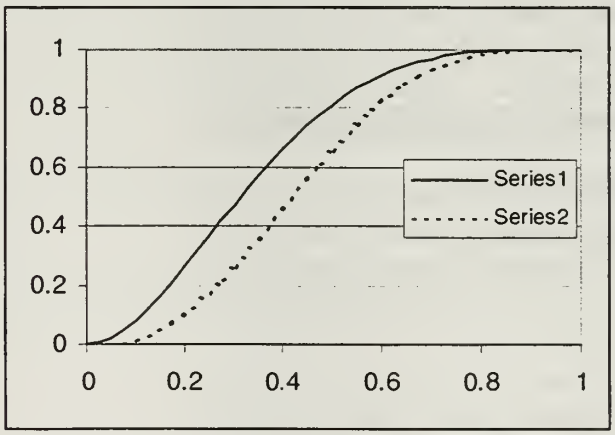


Figure F.3: FSD, Second Sufficient Condition. Series1 dominates Series2

Definition of Second Degree Stochastic Dominance (SSD): SSD is the dominance that can be established by the application of the following definition:

Let f, g be two alternatives with cdf F(x) and G(x) respectively. We say that f dominates g on 2nd degree stochastic dominance (f SSD g), if and only if the area between the two curves is positive.

$(f \text{ SSD } g) \Leftrightarrow \int [F(x) - G(x)]\, dx \geq 0$

Observation

(1) Figure G.3 represents SSD.

(2) Figure G.1 also represents SSD of cdf2 over cdf1 if the area under cdf2 is greater than the area under cdf1.

Definition of sufficient conditions for SSD: FSD is sufficient for SSD.

$(f \text{ FSD } g) \Rightarrow (f \text{ SSD } g)$

Definition of Third Degree Stochastic Dominance (TSD): The third degree of stochastic dominance is the preference for positive skewness on the pdfs. The skewness (γ) is defined as the ratio of the third moment over the standard deviation to the third.

$\gamma = [\int f(x)\,(x - \mu)^3\, dx\,] / \sigma^3$

Definition of the sufficient conditions for TSD:

(1) FSD is sufficient for TSD.

(2) SSD is sufficient for TSD.

LIST OF REFERENCES

(Abdel-Hamid, 1989) Abdel-Hamid, T. *Lessons learned from modeling the dynamics of Software.* Communications of the ACM. December, 1989.

(Abdel-Hamid, 1991) Abdel-Hamid, T. *Software Project Dynamics: An Integrated Approach.* Prentice Hall, 1991.

(Agresti, 1992) Agresti and Evanco. *Projecting Software Defects from Analyzing Ada Designs.* IEEE Transactions on Software Engineering, vol 18 no. 11, November 1992, pp. 988-997.

(AIAA, 1993) American Institute of Aeronautic and Astronautics. *Recommended Practice for Software Reliability*, ANSI/AIAA R-013-1992, February 1993.

(ANSI, 1991) ANSI/IEEE *Standard Glossary of Software Engineering Terminology.* STD-729-1991.

(Albrecht, 1979) Albrecht, A. *Measuring Application Development Productivity.* Proceedings IBM. October 1979.

(Albrecht, 1983) Albrecht, A. and Gaffney, J. *Software Function Source Lines of Code and Development Effort prediction.* IEEE Transactions on Software Engineering, SE-9, 1983.

(Badr, 1993) Badr, S. *A Model and Algorithms for a Software Evolution Control System.* PhD Dissertation, Computer Science Department. Naval Postgraduate School. Monterey, CA. 1993.

(Baligh et al., 1994) Baligh, H., Burton, R., and Obel, B. Validating an Expert System that Designs Organizations. In Computational Organization Theory edited by Carley, K. and Prietula, M. Lawrence Erlbaum Associates, Publishers. 1994.

(Baligh et al., 1996) Baligh, H., Burton, R., and Obel, B. *Organizational Consultant: Creating a Useable Theory for Organizational Design.* Management Science, 42(12). 1996.

(Baybutt, 1989) Baybutt, P. *Uncertainty in Risk Analysis. Mathematics in Major Accidents Risk Analysis.* Edited by R.A. Cox. Clarendon Press - Oxford, 1989.

(Beck, 1999) Beck, K. *Extreme Programming Explained. Embrace Change.* Addison-Wesley, 1999.

(Berzins, 1990) Berzins, V. and Luqi. *Software Engineering with Abstractions.* Addison-Wesley, 1990.

(Boehm, 1981) Boehm, B. *Software Engineering Economics.* Prentice Hall, 1981.

(Boehm, 1984) Boehm, B. *Verifying and Validating Software Requirements and Design Specifications.* IEEE Software, January 1984.

(Boehm, 1988) Boehm, B. *A Spiral Model of Software Development and Enhancement. Computer.* May, 1988.

(Boehm, 1988a) Boehm, B. and Belz, F. *Applying Process Programming to the Spiral Model.* Proceedings of the 4th International Software Process Workshop. May 1988.

(Boehm, 1989) Boehm, B. *Software Risk Management.* IEEE Computer Society Press. 1989.

(Boehm, 1991) Boehm, B. *Software Risk Management: Principles and Practices.* IEEE Software, January, 1991.

(Boehm, 1997) Boehm, B. & De Marco T. *Software Risk Management.* IEEE Software. May-June, 1997.

(Boehm, 2000) Boehm, B., Madachy R., Selby, R. *Cost Models for Future Software Life Cycle Processes: COCOMO 2.0.* http://sunset.usc.edu/COCOMOII/cocomo.html

(Boehm et al, 2000) Boehm, B. et al. Software Cost Estimation with COCOMO II. Prentice Hall, 2000.

(Brooks, 1974) Brooks, F. *The Mythical Man-Month.* Datamation. December. 1974.

(Brown & Eisenhardt, 1998) Brown, S. and Eisenhardt, K. *Competing on the Edge. Strategy as Structured Chaos.* Harvard Business School Press. 1998.

(Burton & Obel, 1998) Burton, R., and Obel, B. *Strategic Organizational Diagnosis and Design. Developing Theory for Application.* Kluwer Academic Publishers. 1998.

(Campbell & Stanley, 1966) Campbell, D. and Stanley, J. Experimental and Quasi-experimental Designs for Research. Rand McNally, 1966

(Carr, 1997) Carr, M. *Risk Management May not be for Everyone.* IEEE Software, May - June 1997.

(Charette, 1997) Charette, R., Adams, K., & White, M. *Managing Risk in Software Maintenance.* IEEE Software, May-June, 1997.

(Chen, 1978) Chen, E. *Program Complexity and Programmer Productivity.* IEEE Trans. Soft. Eng. May 1978.

(Christiansen, 1993) Christiansen, T. R. *Modeling the Efficiency and Effectiveness of Coordination in Engineering Design Teams.* Ph.D. Dissertation, Department of Civil Engineering, Stanford University. Published as Det Norske Veritas Research Report No. 93-2063, Oslo, Norway.

(Chulani et al., 1999) Chulani, Sunita Boehm, Steece. *Bayesian Analysis of Empirical Software Engineering Cost Models.* IEEE Transactions on Software Engineering. July-August, 1999.

(Cohen, 1992) Cohen G.P. *The Virtual Design Team: An Object-Oriented Model of Information Sharing in Project Teams* [Ph.D.]. Stanford: Stanford University, 1992.

(Conklin, 1988) Conklin, J. and Begeman, M. *GIBIS: A Hypertext Tool for Exploratory Policy Discussion.* ACM Transactions on Office Information Systems. Vol. 6. October, 1988.

(Conte, 1986) Conte. S, Dunsmore, H. and Shen, V. *Software Engineering Metrics and Models.* Benjamin Cummings. 1986.

(Cook & Campbell, 1976) Cook, T., Campbell, D. *The Design and Conduct of Quasi-Experiments and True Experiments in Field Settings.* In *Handbook of Industrial and Organizational Psychology.* Rand-McNally. Dunnette (editor). 1976.

(Cullen & Frey, 1999) Cullen, A. and Frey, H. *Probabilistic Techniques in Exposure Assessment. A Handbook for Dealing with Variability and Uncertainty in Models and Inputs.* Plenum Press. 1999.

(Cusumano, 1999) Cusumano, M. and Yoffie, D. *Software Development on Internet Time.* Computer. October, 1999.

(Daft, 1989) Daft, R. *Organization Theory and Design.* West Publishing Co., 1989.

(Dalkey & Helmer, 1963) Dalkey, N and Helmer, O. *An Experimental Application of the Delphi Method to the Use of Experts.* Management Science, 1963, 9, 458-467.

(Devore, 1995) Devore, J. *Probability and Statistics for Engineering and the Sciences.* Duxbury. 1995.

(Dooley, 1994) Dooley, K. and Flor, R. *Success and Failure in Total Quality Management Initiatives.* Proceeding of the Chaos Network, Denver, 1994.

(Elsayed, 1996) Elsayed, E. *Reliability Engineering.* Addison Wesley. 1996.

(Fenton & Pfleeger, 1997) Fenton, N. and Pfleeger, S.L. Software Metrics. A Rigorous & Practical Approach. PWS Publishing Co. 1997.

(Field, 1997) Field, T. *When BAD Things Happen to GOOD Projects.* CIO, 15 October, 1997.

(Gaffney, 1988) Gaffney and Davis. *An Approach to Estimating Software Errors and Availability.* SPC-TR-88-007 version 1.0, March 1988. Proceedings of the Workshop on Software Reliability, July 1988.

(Galbraith, 1977) Galbraith, J. R., *Organization Design. Reading*, MA: Addison-Wesley, 1977.

(Garvey, 1997) Garvey, P., Phair, D. and Wilson, J. *An Information Architecture for Risk Assessment and Management.* IEEE Software, May - June 1997.

(Gemmer, 1997) Gemmer. *Risk Management: Moving Beyond Process.* Computer Vol. 30 Issue 5. May, 1997.

(Gilb, 1977) Gilb, T. *Software Metrics.* Winthrop Publishers, Inc. 1977.

(Gilb, 1988) Gilb, T. *Principles of Software Engineering Management.* Addison-Wesley 1988.

(Goel & Okumoto, 1979) Goel, A and Okumoto, K. *Time-Dependent Error-Detection Rate Model for Software and Other Performance Measures.* IEEE Transactions on Software Reliability. August, 1979.

(Hall, 1997) Hall, E. Managing Risk. *Methods for Software Systems Development.* Addison Wesley, 1997.

(Harn, 1999a) Harn, M., Berzins, V. and Luqi. *Software Evolution via Reusable Architecture.* Proceedings of 1999 IEEE Conference and Workshop on Engineering of Computer-Based Systems. Nashville, Tennesee. March, 1999.

(Harn, 1999b) Harn, M. *Computer-Aided Software Evolution Based on Inferred Dependencies.* Proceedings of Conference on Advanced Information Systems Engineering: 6th Doctoral Consortium. Heidelberg, Germany. June, 1999.

(Harn, 1999c) Harn, M., Berzins, V. and Luqi. *A Dependency Computing Model for Software Evolution.* Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering. Kaiserslautern, Germany. June, 1999.

(Harn, 1999e) Harn, M., Berzins, V. and Luqi. *Computer-Aided Software Evolution Based on a Formal Model.* Proceedings of the 13th International Conference on Systems Engineering. Las Vegas, Nevada. August, 1999.

(Harn, 1999f) Harn, M. *Relational Hypergraph Model.* PhD Disssertation. Naval Postgraduate School. Monterey, California. 1999.

(Huberman & Glance, 1998) Huberman, B. and Glance, N. Fluctuating Efforts and Sustainable Cooperation. Chapter 5 on Prietula, M., Carley, K., Gasser L. *Simulating Organizations. Computational Models for Institutions and Groups.* MIT Press, 1998.

(Humphrey, 1987) Humphrey, W. et al. *A Method for Assessing the Software Capability of Contractors.* CMU/SEI-87-TR-23. 1987.

(Humphrey, 1989) Humphrey, W. *Managing the Software Process.* Addison-Wesley, 1989.

(Ibrahim, 1996) Ibrahim, O. *A Model and Decision Support Mechanism for Software Requirements Engineering.* Ph.D. Dissertation. Naval Postgraduate School. Monterey, California. 1996.

(James, 1996) James, G. E. *Chaos Theory. The Essentials for Military Applications.* Naval War College. The Newport Papers, 1996.

(Johnson, 1994) Johnson, N., Kotz, S., and Balakrishnan N. *Continuous Univariate Distributions. Vol. 1.* Wiley & Sons, 1994.

(Jones, 1994) Jones, Capers. *Assessment and Control of Software Risks.* Yourdon Press Prentice Hall, 1994.

(Jones, 1996) Jones, Capers. *By Popular Demand: Software Estimating Rules of Thumb.* Computer, March 1996.

(Jones, 1997) Jones, Capers. *Table of Languages.* 1997. http://www.spr.com/library/0langtabl.htm

(Jin, 1996) Jin, Y. and Levitt, R. (Department of Civil Engineering, Stanford University). *The Virtual Design Team: A Computational Model of Project Organizations.* Paper to appear in Computational and Mathemetical Organization Theory. 1996.

(Kang et al., 1998) Kang, M., Waisel, L. and Wallace, W. *Team Soar. A Model for Team Decision Making.* Chapter 2 on Prietula, M., Carley, K., Gasser L. *Simulating Organizations. Computational Models for Institutions and Groups.* MIT Press, 1998.

(Karolak, 1996) Karolak, D. *Software Engineering Management.* IEEE Computer Society Press, 1996.

(Kauffman, 1995) Kauffman, Stuart. *At Home in the Universe.* Oxford University Press, 1995.

(Kemerer, 1993) Kemerer, C. *Reliability of Function Points Measurements: A Field Experiment.* Communications of ACM, Vol 36 No 2. 1993.

(Kemerer, 1997) Kemerer, C. *Software Project Management. Readings and Cases.* McGraw-Hill. 1997

(Kitchenham, 1993) Kitchenham, B., Kansala, K. *Inter-item Correlations among Function Points.* First International Software metrics Symposium. IEEE Computer Society Press. 1993.

(Kitchenham, 1997) Kitchenham, B., Linkman, S. *Estimates, Uncertainty, and Risk.* IEEE Software. May-June, 1997.

(Kunz et al., 1998) Kunz, J. C., Tore R. Christiansen, Geoff P. Cohen, Yan Jin, Raymond E. Levitt. *The Virtual Design Team: A Computational Simulation Model of Project Organizations.* Communications of the Association for Computing Machinery (CACM) 41 (11), November, 1998, pp. 84-91.

(Levitt et al., 1994) Levitt, R. E., G. P. Cohen, J. C. Kunz, C. I. Nass, T. Christiansen, and Y. Jin (1994), *The Virtual Design Team: Simulating How Organization Structures and Information Processing Tools Affect Team Performance,* in K. Carley and M. Prietula (Eds.) Computational Organizational Theory, Hillsdale, NJ: Lawerence Erlbaum Associates.

(Levitt, 1999) Levitt, R. *The ViteProject Handbook: A User's Guide to Modelling and Analyzing Project Work Processes and Organizations.* Vité ©. 1999.

(Levitt, 2000) Levitt, R. *VDT Computational Emulation Models of Organizations: State of the Art and the Practice.* Center for Integrated Facility Engineering. Stanford University, 2000.

(Lin, 1998) Lin, Z. *The Choice Between Accuracy and Errors. A Contingency Analysis of External Conditions and Organizational Decision Making Performance.* Chapter 4 on Prietula, M., Carley, K., Gasser L. *Simulating Organizations. Computational Models for Institutions and Groups.* MIT Press, 1998.

(Levy, 1998) Levy, H. *Stochastic Dominance. Investment Decision Making under Uncertainty.* Kluwer Academic Publishers. 1998.

(Londeix, 1987) Londeix, B. *Cost Estimation for Software Development.* Addison-Wesley, 1987.

(Lorenz, 1995) Lorenz, Kidd. *OO Software Metrics.* Prentice Hall, 1995.

(Luqi, 1988a) Luqi and Ketabchi, M. *A Computer-Aided Prototyping System.* IEEE Software. March, 1988.

(Luqi, 1988b) Luqi and Berzins, V. *Rapidly Prototyping Real-Time Systems.* IEEE Software. September, 1988.

(Luqi, 1989) Luqi. Software Evolution *Through Rapid Prototyping.* IEEE Computer. May, 1989.

(Luqi, 1990) Luqi. *A Graph Model for Software Evolution.* IEEE Transactions on Software Engineering. Vol. 16 No. 8. August, 1990.

(Luqi) Luqi. *Formal Models and Prototyping.* Research supported by National Science Foundation (CCR-9058453) and by the Army research Office (30989-MA).

(Luqi, 1991) Luqi and Royce, W. *Status Report: Computer-Aided Prototyping.* IEEE Software. November, 1991.

(Luqi, 1997) Luqi and Goguen, J. *Formal Methods: Promises and Problems.* IEEE Software. January, 1997.

(Lyu, 1995) Lyu, M. *Software Reliability Engineering.* IEEE Computer Society Press. 1995.

(McFarlan, 1974) McFarlan, F. *Portfolio Approach to Information Systems.* Harvard Business Review. January-February, 1974.

(Marshall, 1995) Marshall, K. Oliver, R. *Decision Making and Forecasting.* McGraww-Hill, 1995.

(Mostov, 1989) Mostov, Luqi and Hefner. *A Graph Model of Software Maintenance.* Technical Report NPS52-90-014. Department of Computer Science. Naval Postgraduate School. Monterey, CA. August 1989.

(Mostov, 1989) Mostov. *A Model of Software Maintenance for Large Scale Military Systems.* Master's Thesis. Naval Postgraduate School. Monterey, CA. June, 1990.

(Munson, 1995) Munson, J. and Khoshgofar, T. Chapter 12 (Lyu, 1995).

(Musa, 1998) Musa, J. *Software Reliability Engineering: More Reliable Software, Faster Development and Testing.* McGraw-Hill, 1998.

(Myers, 1976) Myers, G. *Software Reliability.* John Wiley & Sons. 1976.

(Nissen, 1998) Nissen, M. *Redesigning Reengineering through Measurement-Driven Inference.* MIS Quarterly. December, 1998.

(Nogueira et al., 2000a) Nogueira, J.C., Luqi, and Berzins, V. *A Formal Risk Assessment Model for Software Evolution.* SEKE 2000. Chicago, July 2000.

(Nogueira et al., 2000b) Nogueira, J.C., Luqi, and Bhattacharya, S. *A Risk Assessment Model for Software Prototyping Projects.* IEEE Workshop on Rapid System Prototyping RSP 2000. Paris, June 2000.

(Nogueira et al., 2000c) Nogueira, J.C., Luqi, , Berzins, V., and Nada, N. *A Formal Risk Assessment Model for Software Evolution.* ICSE 2000. Limerick, June 2000.

(Nogueira et al., 2000d) Nogueira, J.C., Luqi, and Berzins, V. *Risk Assessment in Software Requirement Engineering.* IDTP 2000. Dallas, June 2000.

(Nogueira et al, 2000e) Nogueira, J.C., Jones, C. R., and Luqi. *Surfing on the Edge of Chaos: Applications to Software Engineering.* CCRP 2000. Monterey, June 2000.

(Norden, 1963) Norden, Peter. *Resource Usage and Network Planning Techniques.* In Operations Research in Research and Development. Edited by Dean, B. John Wiley & Sons 1963 pp. 149-169.

(O'Leary, 1988) O'Leary, D. *Methods of Validating Experts Systems.* Interfaces #18. 1988.

(Pearl, 2000) Pearl, J. *Causality. Models, Reasoning and Inferrence.* Cambridge University Press, 2000.

(Porter, 1980) Porter, Michael. *Competitive Strategy.* Free Press, 1980.

(Pfleeger, 1999) Pfleeger, S.L. *Albert Einstein and Empirical Software Engineering.* IEEE Computer. October 1999.

(Pressman, 1992) Pressman. *Software Engineering*, 1992.

(Prietula et al, 1998) Prietula, M., Carley, K., Gasser L.Simulating Organizations. Computational Models for Institutions and Groups. MIT Press, 1998.

(Putnam, 1980) Putnam, L. *Software Cost Estimating and Life-cycle Control: Getting the Software Numbers.* IEEE Computer Society Press. 1980.

(Putnam, 1992) Putnam, L. and Myers, W. *Measures for Excellence. Reliable Software On Time Within Budget.* Yourdon Press, 1992.

(Putnam, 1996) Putnam, L. and Myers, W. *Executive Briefing. Controlling Software Development.* IEEE Computer Society Press. 1996.

(Putnam, 1997) Putnam, L. and Myers, W. *Industrial Strength Software.* Effective Management Using Measurement. IEEE Computer Society Press, 1997.

(Ramesh, 1992) Ramesh, B. and Dhar, V. *Supporting Systems Development Using Knowledge Captured During Requirements Engineering.* IEEE Transactions on Software Engineering. June, 1992.

(Ramesh, 1995) Ramesh and Luqi. *An Intelligent Assistant for Requirements Validation.* Journal of Systems Integration, 5, 157-177. 1995.

(Reel, 1999) Reel, J. *Critical Success Factors in Software Projects.* IEEE Software. May - June, 1999.

(Render, 1997) Render, B. and Stair, R. *Quantitative Analysis for Management.* Prentice Hall, 1997.

(Rifkin, 2000) Rifkin, S. *When the Project Absolutely Must Get Done: Marrying the Organization Chart with the Precedence Diagram.* International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, June 2000.

(Reliasoft, 2000) Weibull++ Ver 5.0. Reliasoft. http://www.reliasoft.com

(Roberts, 2000) Roberts, N. *Coping with Wicked Problems.* Third BI-Annual Research Conference of the International Public Management Network. Sydney, March 2000.

(Rome Lab, 1992) Rome Laboratory. *Methodology for Software Reliability Prediction and Assessment.* Technical Report RL-TR-92-52. 1992

(Roos, 1996) Roos, Johan. *The Poised Organization: Navigating Effectively on Knowledge Landscapes.*, 1996. http://www.imd.ch/fac/roos/paper_po.html

(Santosus, 1998) Santosus, Megan. *Simple, Yet Complex.* Business Management CIO Enterprise Magazine. April 15, 1998.

(SEI, 1996) Software Engineering Institute. *Software Risk Management.* Technical Report CMU/SEI-96-TR-012. June, 1996.

(Schneidewind, 1975) Schneidewind, N. *Analysis of Error Processes in Computer Software.* Proceedings of the International Conference on Reliable Software. IEEE Computer Society, 21-23 April 1975. Pp 337-346.

(Senegupta and Jones, 1999) Sengupta, K. and Jones Carl R. *Creating Structures for Network-Centric Warefare: Perspectives from Organizational Theory.* Command & Control Research & Technology Symposium. CCRP 1999. Naval War College, 1999.

(Sommerville, 1992) Sommerville, I. *Software Engineering*. 1992

(Thomsen et al., 1999) Thomsen, Jan, Raymond E. Levitt, John C. Kunz, Clifford I. Nass, Douglas B. Fridsma. *A Trajectory for Validating Computational Emulation Models of Organizations.* Journal of Computational & Mathematical Organization Theory, 5, (4), December 1999, pp. 385-401

(Turban & Aronson, 1998) Turban, E. and Aronson, J. *Decision Support Systems and Intelligent Systems.* Prentice Hall. 1998.

(USAF, 1988) USAF. *Software Risk Abatement.* ASFC/AFLC pamphlet 800-45, US Air Force Systems Command. Andrews AFB. 1988.

(USC, 2000) University of South California. The Win Win Spiral Model and Groupware Support System http://sunset.esc.edu/research/WINWIN/winwin_main.html 2000.

(vanGenuchten, 1991) van Genuchten, M. *Why is Software Late? An Empirical Study of the Reasons for Delay in Software Development.* IEEE Transactions on Software Engineering. June, 1991.

(von Bertalanfy, 1976) von Bertalanfy, L. *General System Theory: Foundations, Development, Applications.* Braziller, 1976.

(Walston, 1977) Walston, C. and Felix, C. *A Method of Programming Measurement and Estimation.* IBM Systems Journal. Vol. 16 No. 1, 1977.

(Weibull, 1939) Weibull, W. *A Statistical Theory of the Strength of Material. Report No. 151*, Ingeniors Vetenskaps Akademiens Handligar. Stockholm, 1939.

(Whitmore & Findlay, 1978) Whitmore, G. and Findlay, M. *Stochastic Dominance.* Lexington Books. 1978.

(Wideman, 1992) Wideman, R. *Risk Management. A Guide to Managing Project Risk Opportunities.* Project Management Institute. 1992.

(Woodward, 1965) Woodward, J. *Industrial Organization Theory and Practice.* Oxford University Press. 1965.

(Woodward, 1999) Woodward, S. *Evolutionary Project Management.* IEEE Computer, October 1999.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0094
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5000

3. Dr. Dan Boger..... 1
   Naval Postgraduate School
   Code CS/Bo
   411 Dyer Rd.
   Monterey, California 93943-5000

4. Dr. Luqi 5
   Naval Postgraduate School
   Code CS/Lq
   411 Dyer Rd.
   Monterey, California 93943-5000

5. Dr. Valdis Berzins..... 1
   Naval Postgraduate School
   Code CS/Be
   411 Dyer Rd.
   Monterey, California 93943-5000

6. Dr. Carl Jones..... 1
   Naval Postgraduate School
   Code IS/Js
   411 Dyer Rd.
   Monterey, California 93943-5000

7. Dr. Man-Tak Shing..... 1
   Naval Postgraduate School
   Code CS/Sh
   411 Dyer Rd.
   Monterey, California 93943-5000

8. Dr. Mark Nissen..... 1
   Naval Postgraduate School
   Code SM/Ni
   411 Dyer Rd.
   Monterey, California 93943-5000

9. Dr. Swapan Bhattacharya.....1
   Dept. of Computer Science & Engineering
   Jadavpur University
   Calcutta – 700032
   India

10. Lawrence H. Putnam Sr..... 1
    QSM, Inc.
    2000 Corporate Ridge, Suite 900
    McLean, VA 22102

11. Comando General de la Armada 5
    Edificio Comando General 4 Piso
    Rambla 25 de Agosto de 1825 S/N
    Montevideo, CP 11000
    Uruguay

12. CAPT Juan C. Nogueira..... 5
    Edificio Comando General 4 Piso
    Rambla 25 de Agosto de 1825 S/N
    Montevideo, CP 11000
    Uruguay